

2

AD-A277 024



CDRL No. 0002AD

Final Report For The Manufacturing Optimization (MO) System

Linda J. Lapointe
Thomas J. Laliberty
Robert V.E. Bryant

Raytheon Company

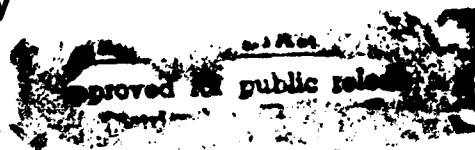
1994



13106 94-08411

ARPA

Advanced Research
Projects Agency



DTIC QUALITY INSPECTED 1

94 3 15 015

CDRL No. 0002AD

Final Report For The Manufacturing Optimization (MO) System

Prepared by
Linda J. Lapointe
Thomas J. Laliberty
Robert V.E. Bryant

Raytheon Company
Missile Systems Laboratories
Tewksbury, MA 01876

February 1994

ARPA Order No. 8363/02
Contract MDA972-92-C-0020

Prepared for

ARPA
Advanced Research
Projects Agency

Contracts Management Office
Arlington, VA 22203-1714

Accession For	
NTIS	CRA&I <input checked="" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification <i>not A273345</i>	
By _____	
Distribution/ _____	
Availability Codes	
Dist	Avail and/or Special
<i>A-1</i>	

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE 28 February 94	3. REPORT TYPE AND DATES COVERED Final		
4. TITLE AND SUBTITLE Final Report For The Manufacturing Optimization (MO) System		5. FUNDING NUMBERS (C) MDA972-92-C-0020		
6. AUTHOR(S) Robert V. E. Bryant, Thomas J. Laliberty and Linda J. Lapointe				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Raytheon Company 50 Apple Hill Drive Missile Systems Laboratories Tewksbury, MA 01876		8. PERFORMING ORGANIZATION REPORT NUMBER CDRL 0002AD		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Advanced Research Projects Agency (ARPA) Contracts Management Office Arlington, VA 22203-1714		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT		12b. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words) This is the final report for the Manufacturing Optimization (MO) System. The purpose of the MO system is to enable all manufacturing specialists to participate in the product/process development activity concurrently. The system consists of a set of tools to model the manufacturing processes and centralize the various process tradeoffs. Recommendations can be compared and negotiated among the individual manufacturing participants. After the manufacturing team has reached a consolidated position, the results are passed back to the cross functional (top level) team for their negotiation.				
14. SUBJECT TERMS DFMA, Process Modeling, Enterprise Modeling, MO		15. NUMBER OF PAGES 130		
		16. PRICE CODE		
17. SECURITY CLASSIFICATION OF THIS REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Contents

1	Summary	1
2	Introduction	3
2.1	Subject.....	3
2.1.1	The Role of DICE-MO in CE.....	4
2.1.2	Product / Process Development	9
2.2	Scope	10
2.4	Document Plan	11
3	Methods, Assumptions and Procedures	12
3.1	Evaluation of Existing CE Technology	12
3.2	Collaborative DFMA Tool	12
3.3	Prototype Development Methodology	13
3.3.1	Product and Enterprise Information Modeling	15
3.3.2	Object-Oriented Analysis.....	16
3.3.3	Object-Oriented Design.....	16
3.3.4	Object-Oriented Programming.....	16
3.3.5	Testing and Implementation	17
3.3.6	Maintenance	17
4	Results and Discussion	18
4.1	CE Technology Evaluation.....	18
4.2	Information Modeling	19
4.3	Prototype System Architecture	20
4.4	Prototype System Components.....	24
4.4.1	Product (STEP) Models.....	24
4.4.2	Process Models	25
4.4.3	Manufacturing Analyzer.....	27
4.4.3.1	Process Analyzer	27
4.4.3.2	Yield & Rework Analyzer	28
4.4.3.3	Cost Estimator	28
4.4.4	Manufacturing Advisor.....	29
4.4.6	Process Modeler	30
4.4.7	RAPIDS.....	32
4.5	Test Cases.....	33
4.5.1	PWB Fabrication and Assembly	33

	4.5.1.1 PWB Assembly Model.....	33
	4.5.1.2 PWB Fabrication Model	36
	4.5.1.3 PWA Analysis Report.....	38
	4.5.1.4 PWF Analysis Report.....	40
	4.5.3 MCM Fabrication	43
	4.6 How To Build a New Product/Process Model	43
5	Conclusions.....	46
6	Recommendations.....	48
	6.1 Introduction.....	48
	6.2 DICE MO Software Enhancements	48
	6.2.1 User Interface - Functions and Presentation.....	48
	6.2.2 Manufacturing Process Modeler/Analyzer Enhancements	49
	6.2.3 Integration and Interfaces	49
	6.3 Research and Development Recommendations.....	50
7	References.....	51
8	Notes.....	52
	8.1 Acronyms	52
1	Manufacturing Enterprise Model	53
	1.1 Process Information Model	53
	1.1.1 EXPRESS Schema for Process Model.....	53
	1.1.1.1 ProcessModel Entity	54
	1.1.1.2 Process Entity.....	54
	1.1.1.3 Operation Entity.....	56
	1.1.1.4 Scrap Entity	57
	1.1.1.5 Rework Entity.....	57
	1.1.1.6 OpCost Data.....	58
	1.1.2 EXPRESS-G Schema for Process Model	59
	1.2 Resource Information Model	60
	1.2.1 EXPRESS Schema for Resource	60
	1.2.1.1 ResourceUtilization Entity.....	60
	1.2.1.2 Resource Entity	61
	1.2.1.3 Parameter Entity	61
	1.2.1.4 Labor Entity	62
	1.2.1.5 Equipment Entity	62
	1.2.1.6 Facility Entity	62
	1.2.1.7 ConsumableMaterial Entity.....	63

I.2.2	EXPRESS-G Schema for Resource.....	64
I.3	Selection Rules Information Model.....	64
I.3.1	EXPRESS Schema for Selection Rules.....	64
I.3.1.1	Constants and Types for Rule Construction.....	66
I.3.1.2	DataDictStr Entity.....	67
I.3.1.3	ComplexRule Entities.....	68
I.3.1.4	Expression Entities.....	68
I.3.1.5	Equation Entities.....	69
I.3.1.6	Term Entities.....	69
I.3.2	EXPRESS-G Schema for Selection Rules.....	71
II	PWB Product Information Model.....	72
II.1	Printed Wiring Board Product Data Model.....	72
II.1.1	PWB Design Schema.....	73
II.1.2	PWB Generic Types and Entities.....	75
II.1.3	Header Data Schema.....	76
II.1.4	Alias Data Schema.....	77
II.1.5	Annotation Data Schema.....	77
II.1.6	CARI Data Schema.....	78
II.1.7	Class Data Schema.....	78
II.1.8	Comment Data Schema.....	79
II.1.9	Design Rule Data Schema.....	79
II.1.10	Gate Data Schema.....	83
II.1.11	Net Data Schema.....	83
II.1.12	Metal Area Data Schema.....	85
II.1.13	Part Data Schema.....	86
II.1.14	Pin Data Schema.....	89
II.1.15	Conductor Routing Data Schema.....	89
II.1.16	Via Data Schema.....	90
II.1.17	Library Cross Reference Data Schema.....	91
II.2	PWB Design Data EXPRESS-G Model.....	92
II.3	Electronic Component Library Data Model.....	92
II.3.1	Component Model Data Schema.....	92
II.3.2	Pad Stack Data Schema.....	95
II.3.3	Pad Shape Data Schema.....	95
II.4	Electronic Component Library Data EXPRESS-G Model.....	96
III	User Interface Screens.....	97

III.1	File Menu	97
III.1.1	Product/STEP Data Selection.....	98
III.1.2	Process Model Selection	98
III.1.3	RAPIDS to STEP Translator Interface	99
III.1.4	STEP to RAPIDS Translator Interface	100
III.1.5	STEP Editor	100
III.2	Analyzer Form.....	100
III.3	Advisor Window.....	101
III.3.1	File.....	101
III.3.2	View	102
III.3.3	Graphs	103
III.3.3.1	Activity Graphs	103
III.3.3.2	Quality Graphs.....	104
III.3.3.3	Costing Graphs	105
III.3.4	Reports.....	108
III.4	Modeler Window.....	110
III.4.1	Manufacturing Activity Node Definition	112
III.4.2	Selection Rules Definition	113
III.4.2.1	Yield Rate Definition	114
III.4.2.2	Rework Rate Definition.....	115
III.4.3	Resource Definition.....	116

Figures

Figure 2-1. Two Level Team Concept	5
Figure 3-2. Intelligent Product, Process and Resource Models.....	10
Figure 3-2. DICE-MO Prototype Development Methodology.....	14
Figure 4-1 Sample PWB Design Cycle Flow	19
Figure 4-2 MO System Architecture	21
Figure 4-3 Process Modeler Block Diagram.....	22
Figure 4-4 Manufacturing Analyzer Block Diagram.....	23
Figure 4-5 Manufacturing Advisor Block Diagram.....	23
Figure 4-6 Printed Wiring Assembly Process Model.....	27
Figure 4-7 Sample Yield versus Process Comparison Graph	30
Figure 4-8 Process Modeler User Interface Window.....	32
Figure 4-9 Printed Wiring Board Assembly Tree Model.....	34
Figure 4-10 Printed Wiring Board Assembly Flow	34
Figure 4-11 Process PWA Activity Specification.....	35
Figure 4-12 Process PWA Selection Rules.....	35
Figure 4-13 Operation 270 Activity Specification & Selection Rules.....	35
Figure 4-14 Printed Wiring Board Fabrication Tree Model.....	36
Figure 4-15 Printed Wiring Board Fabrication Flow	37
Figure 4-16 Process PWF Activity Specification & Selection Rules	37
Figure 4-17 Operation 130 Activity & Yield Specifications	37
Figure I-1 EXPRESS-G Model of Process Model Schema	59
Figure I-2 EXPRESS-G Model of Resources Schema.....	64
Figure I-3 EXPRESS-G Model of Selection Rules Schema.....	71
Figure II-1 PWB Schema Level EXPRESS-G Model	92
Figure II-2 Component Data EXPRESS-G Schema.....	96
Figure III-1 MO Main Window	97
Figure III.1 File Options.....	97
Figure III.1.1 Product Data Selection/Edit Menu	98
Figure III.1.2 Process Model Selection Menu	99
Figure III.1.3 RAPIDS to STEP Form.....	100
Figure III.1.4 STEP to RAPIDS Form.....	100
Figure III.3 Manufacturing Advisor Window	101
Figure III.3.1 File menu	102

Figure III.3-2 View menu.....	102
Figure III.3.3.1 Process Plan List View Form.....	104
Figure III.3.3.2-1 Quality Graphs	105
Figure III.3.3.2-2 Yield versus Process Graph.....	105
Figure III.3.3.3-1 Costing Graphs	106
Figure III.3.3.3-2 Actual Time versus Process Graph	107
Figure III.3.3.3-3 Quantity Value for an Exact Point.....	108
Figure III.3.4-1 Analysis Reports Form	109
Figure III.4-1 Process Modeler Window	111
Figure III.4-2 Process Model Selection Window.....	112
Figure III.4.1 Manufacturing Activity Specification Window.....	113
Figure III.4.2-1 Selection Rules Window	113
Figure III.4.2-2 Rule Specification.....	114
Figure III.4.2.1 Yield Specification Window	115
Figure III.4.2.2 Rework Specification Window.....	116
Figure III.4.3-1 Resource Utilization Window	117
Figure III.4.3-2 Resource Window	117
Figure III.4.3-3 Resource Specification Window	118
Figure III.4.3-4 Facility Resource Specification Window.....	118
Figure III.4.3-5 Equipment Resource Specification Window.....	119
Figure III.4.3-6 Resource/Consumable Specification Window.....	119
Figure III.4.3-7 Labor Resource Window.....	120
Figure III.4.3-8 Labor Rate Resource Specification Window.....	120

Tables

Table 2-1 Relation of Design Feature vs. Process Yield	9
Table 3-1 Software Methods and Tools used in Prototype Development	14

1 Summary

This is the Final Report for the DICE - Manufacturing Optimization (DICE-MO) program. The mission of the DICE-MO program was to investigate the viability of supporting a two-tiered product design and manufacturing process development team. The approach enables refinement of the product design and manufacturing process through collaboration of product design and process development experts.

This investigation had two primary goals. First, performing new research and development of enabling technologies in Design For Manufacturing and Assembly (DFMA). This R&D effort was centralized around the development of a DFMA analysis tool. The MO prototype system developed analyzes a unified product model represented in STEP against an established manufacturing enterprise model. The manufacturing enterprise model captures an abstract representation of the manufacturing processes and resources supported by the enterprise. The enterprise model supports representation of reasoning logic which directly associates features of the product model to the processes and resources of the enterprise. The analysis tool evaluates the reasoning logic and selects the appropriate processes and resources. Captured historical cost, yield and rework data from these processes or similar processes enable the system to estimate the cost, yield and rework rates for manufacturing the product under evaluation.

The second goal of the program was to "user harden" existing concurrent engineering technology. The following three tools developed under the DICE program were evaluated for inclusion in the MO system: 1) The Requirements Manager (RM), a system designed to manage product requirements, specifications and corporate policies to support concurrent engineering; 2) The Project Coordination Board (PCB) which provides support for the coordination of the product development activities in a cooperative environment; 3) The Rensselaer Object System for Engineering (ROSE) an object data manager that has been developed for engineering applications and enhanced to support the DICE program.

Raytheon planned on using ROSE to store and manage the data files for the manufacturing processes, operations, and steps, as well as, the various manufacturing analysis results. The PCB was to be used for monitoring the overall product design cycle activities. The RM was to be used to model and evaluate the manufacturability/producibility guidelines.

The ROSE DB was evaluated and found to be suitable for the proposed task. All data used by the MO prototype is stored and managed within ROSE. The RM was evaluated as a stand-alone system. Raytheon determined that the system was capable of modeling and evaluating the manufacturing guidelines. The plan was to implement a two way interface between MO and the RM using the RM API. Although planned, the RM API never supported two way communication, only an export function. Raytheon believes that a two way interface would be advantageous to the RM. The PCB was evaluated as a means to support the communication of the product/process development activities within the product design cycle. A simple model was entered into the PCB as a means of evaluating its viability. The PCB proved too immature a software product to be of added value to the MO system.

The primary output of the program was a prototype software system that is capable of analyzing a product model and a manufacturing process model which would provide a preliminary list of the manufacturing processes and estimates of the the cost and yield to produce the product. The ROSE database from STEP Tools Inc. was the only existing concurrent engineering technology which we evaluated that was included in the final software product.

2 Introduction

2.1 Subject

Traditionally, manufacturability or producibility analysis on a new design is performed by one or more representatives from the manufacturing operation depending on the type of product. For simple machined piece parts it may be just one representative, while for more complex electro-mechanical assemblies there may be several manufacturing engineers involved, each focusing on the part of the design that involves his specialty. The method of analysis varies throughout industry. Methods range from ad-hoc reviews to formal procedures involving checklists, cost analysis and grading systems.

The purpose of design for manufacture and assembly is to get early insight into the manufacturing requirements and cost of a new design. Manufacturability analysis must be performed early to have a significant impact on design. However, the ability to perform a meaningful analysis early in the design cycle is often affected by the product type. For instance, early on in the design of a PWB the form factor is established. This describes the overall dimensions of the PWB. The overall size and shape of the board may set a limiting requirement on the maximum number of layers that may be used. As more details of the design become resolved (e.g., trace width, pad sizes, etc.) we can estimate more accurately the overall cost and manufacturability of the PWB. Further in the design cycle, detailed schematics will be developed and components are selected, each of these gives more information about the design and the ultimate cost of the board. Therefore, we must iterate through several manufacturing analyses throughout the design cycle. The estimates made by the analysis will continually be refined and become more accurate as more details are known. The timing of analysis varies according to product types and their specific design cycles.

The MO system is a general purpose enabling technology for design for manufacture and assembly (DFMA) system that can be used for an array of product types. The system will enable one or a number of engineers to perform manufacturability analyses, merge and compare their results. The system consists of the following three integrated modules: a manufacturability analysis tool that analyzes a product model described in STEP against an enterprise model consisting of process and resource capabilities of the manufacturing

enterprise, an advisor tool that presents the results of the analysis runs in both graphical and textual formats, and an enterprise modeler which provides the process engineer the means to capture his process flow logic and resource utilization in terms of product features.

2.1.1 The Role of DICE-MO in CE

The DICE program has been chartered with developing technology that enables concurrent engineering. The DICE concurrent engineering model is a replication of the human tiger team concept that has been successfully used on small scale projects to bring high quality products to market quickly. The basic tenet of the human tiger team is to have the various specialists contributing to the project co-located. In today's environment of complex product designs and geographically dispersed specialists, DICE envisions a "virtual tiger team" working on a "unified product model" accessible by computer networks. Such an environment must enable the specialist from each functional area to work on the design concurrently and share development ideas.

Raytheon proposes a conceptual refinement to the DICE virtual tiger team concept. In the present DICE virtual tiger team model, all functions are represented and linked concurrently to the product design at a single level. We propose a two level approach with a virtual product team having a global view supported by information supplied by the virtual process teams. See Figure 2-1. The rationale for this refinement is based on the growing complexity of both the products and supporting development processes. It is increasingly difficult to have one representative adequately support a manufacturing (or logistics) position that involves numerous specialized process areas. In practice, the assigned representative is usually a specialist in one of the process areas and has only generalized knowledge about the other processes. The "virtual process team" concept would enable representation from all the process areas. The product team would concentrate on using the cost/yield information supplied to them by the process teams and determining the best plan taking into account the existence or implementation of manufacturing, logistics, or test plans. The product team would be responsible for decisions which span cross-functional expertise.

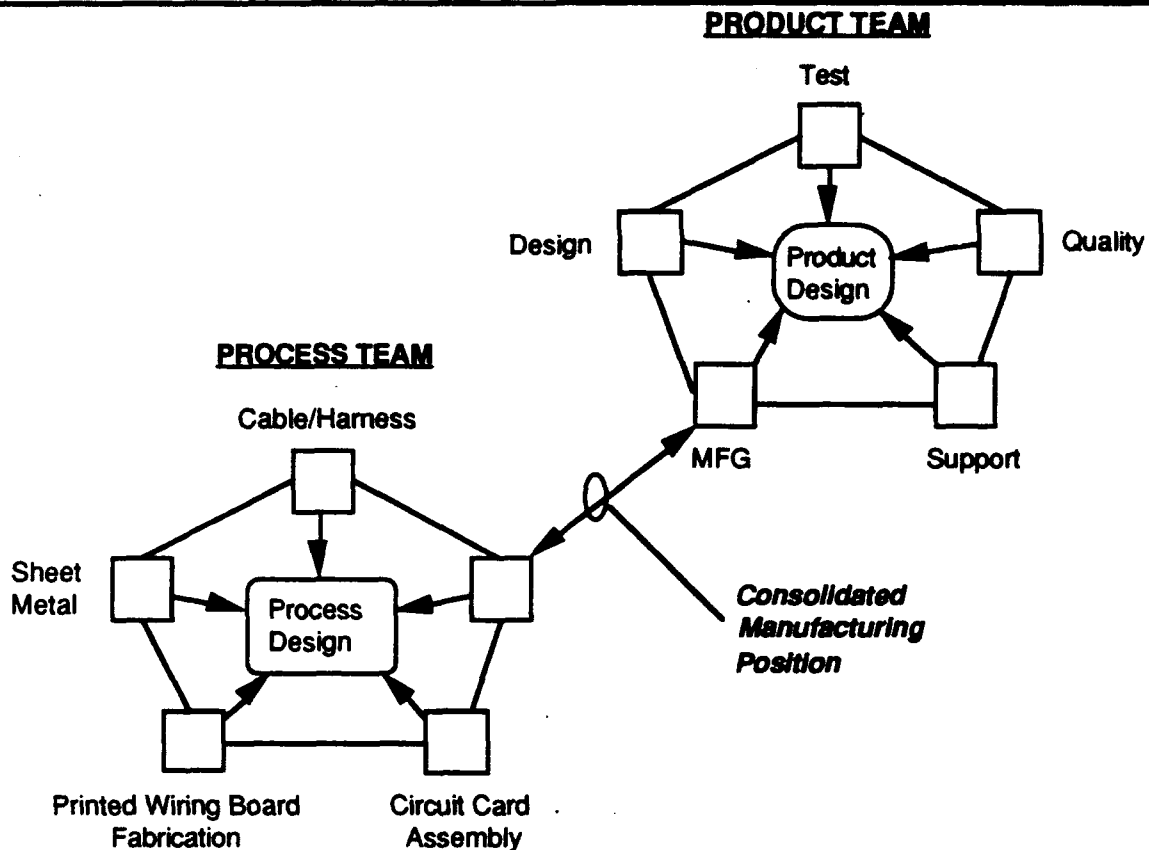


Figure 2-1. Two Level Team Concept

Ideally, the virtual process team is an extension of the product team. It will consist of specialists representing all the various processes. For instance, a process team might consist of a PWB Fabrication, Circuit Card Assembly, Cable/Wire Harness, and Sheet Metal representative. They will have access to the unified product database and will be responsible for the manufacturing inputs to the product team. Each member of the process team will review the design, perform a DFMA analysis, and make design or process change recommendations. The product team will then negotiate with the process team to arrive at a position (and perhaps alternatives) consistent with the manufacturing plan.

An alternate approach to the two levels of teams is to have all the specialists participate in the product virtual team. However, for reasonably complex products/processes this practice is inefficient because it does not take into account a manufacturing plan for managing the production assets. This plan must include capital and process improvements, quality goals, manufacturing systems and vendor management. Moreover, just knowing the cost drivers from each of the specialized areas does not explain how they will interact

when applied to the design. For instance, will enlarging pads to alleviate drilling problems then cause etching and soldering problems?

The virtual process team will be supported by a set of tools that implements a concurrent DFMA system. These tools will enable the manufacturing process team to perform individual DFMA analyses, merge and review these analyses, and negotiate trade-offs among the processes. The system, Manufacturing Optimization (MO), has the following capabilities:

- 1) Process Analysis - This performs the initial analysis on the design to determine the manufacturing process required to produce the part. We propose a knowledge based approach that will compare design features and attributes against process capabilities to determine a part's process sequence.
- 2) Yield/Rework Estimator - This calculates yield and rework values by performing a look up of design features versus an operation. If multiple features affect an operation then a composite value is computed based on defined relationships among features.
- 3) Cost Estimator - Costs will be calculated for each operation used to produce the part. The individual operation cost will be calculated through resource utilization estimates based on labor standards, wage rates, and production efficiencies. These costs will then be factored by yield and rework rates to arrive at an estimate.
- 4) Manufacturing Advisor - The manufacturing advisor provides the user with various methods to view the results produced from the analyzer. The results can be viewed graphically or textually. The reporting capability allows the user to customize a detailed report which can be printed to the screen or to an ASCII file. MO allows the user to view one or more sets of analysis results at a time. By selecting multiple analysis runs to graphically display, the user can visually compare the analyses.
- 5) Enterprise Modeler - The Enterprise modeler provides a graphical user interface environment to enable defining the process flow and resource utilization.

Selection rules are defined which tie processes and resources directly to product features.

The first step in performing a Design for Manufacturability study is to evaluate the current state of the design. This evaluation provides the baseline for the recommendations and enhancements that lead to a design that is optimized for manufacturing. The proposed evaluation method is based on determining the manufacturing process that will be employed to produce the part, characterizing that process in terms of its cost and predicted yields.

The MO approach is based on modeling relationships between design features and manufacturing processes. That is, we provide a means for linking attributes of a design to the manufacturing process that is ultimately used to produce the part. Characterizing the problem in these terms, highlights its similarities to generative or intelligent process planning problems. Raytheon's experience in both generative process planning and Design for Manufacture has shown that the knowledge needed to perform each task is essentially the same.

The key to performing an accurate manufacturability analysis is to characterize the process that will be used to fabricate the part. This characterization takes the form of a manufacturing process selection algorithm and representation. Basing manufacturing cost analysis on a detailed description of the process will give us visibility into the relationship between the design attributes and the manufacturing process. This will allow us to focus on manufacturing cost drivers and their causes. By characterizing the process in this manner, we are able to review the process that will be used to produce the product and be readily able to consider alternative manufacturing processes and their consequences.

Following this logic, it makes sense to capture the expert's process planning knowledge into a process selection model so that the relationship between the product features and the process selected to fabricate the product is explicitly defined. This does not mean that there is a one to one relationship between the design feature and the process steps. In some areas, such as PWB, the design may be implemented using different technologies, each of which implies a certain process, such as surface mount versus through hole technology. In other cases, there are multiple processes that can be used to produce the same feature. This

is most prevalent in the metal fabrication (machining) area where often a number of processes (investment casting, milling) are capable of producing the part.

There were two development challenges: modeling the information required to represent the manufacturing process such that it can be used for selection and cost estimating, and building a selection logic algorithm that adequately represents the planning logic employed by expert process planning.

One of the major manufacturing cost drivers is production yields. Poor yields increase costs while tying up production capacity and other resources such as engineering support and materials. In prior Raytheon DFMA systems the ability to model production yields and relate these yields to product features has enabled design teams to perform design trade-offs with the understanding of the resulting cost and quality.

Raytheon and industry data has shown that in PWB fabrication, manufacturing yields can be directly tied to design features. Table 2-1 shows some of the relationship between trace width and etching yield, annular ring and drilling yield, and layer count and lamination yield.

Table 2-1. Relation of Design Feature vs. Process Yield

line width(mils)	3	5	8	1
etch yield	85%	92%	97%	99%
layer count	4	8	14	22
lamination	99%	98%	95%	88%
Aspect Ratio	7	4.5	3	
Plating yield	95%	98%	99%	

Each process used in manufacturing requires some set of resources. It is possible to connect a set of resources to a particular process and provide a rate at which those resources are used/consumed. By capturing resource utilization on a per process basis, and providing a means of specifying a product feature based function to calculate the rate of usage. For example, the usage rate of a particular drill press may be dependent on the number of holes of a specific range that exist on a part. In other words, a function could be specified in terms of the number of holes.

2.1.2 Product / Process Development

The use of intelligent interacting product, process and resource models (see Figure 2-2) provide the foundation for supporting computer applications capable of measurement, analysis, planning and synthesis for a number of enterprise functions. The DICE-MO prototype is targeted at analysis and planning activities.

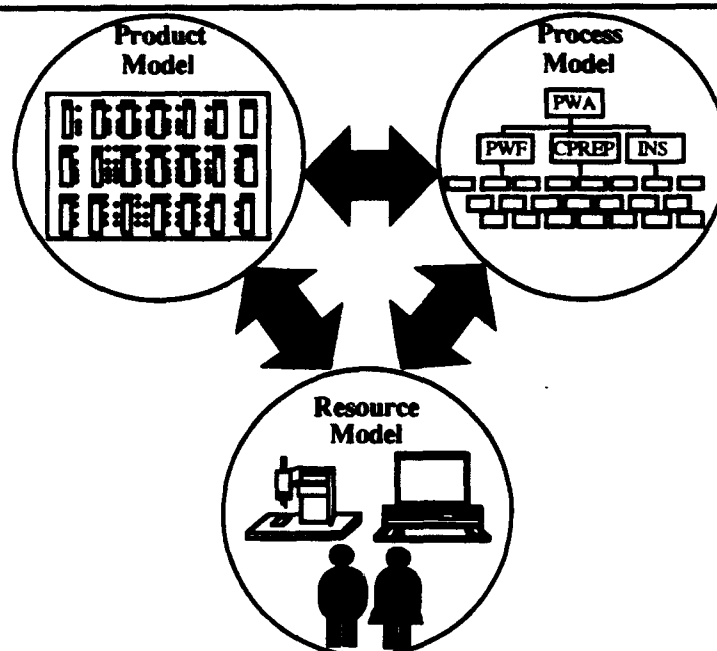


Figure 3-2. Intelligent Product, Process and Resource Models

The ability to reason about the product features and the available resources while planning the manufacturing process allows the process engineer the ability to finely tune the production process. This will ultimately produce more efficient processes. Conversely, providing the designer with insight into the manufacturing process and resources at hand will enable fine tuning of the design for manufacturability and producibility.

The DICE-MO prototype was designed with these notions in mind. The Modeler component of the prototype brings product, process and resource models together in a flexible and consistent manner. Since DICE-MO is intended for use primarily as a manufacturing analysis tool, the perspective taken by the Modeler is that of the process. Emphasis is put on defining processes while having the ability to reference product features and manufacturing resources.

2.2 Scope

This investigation had two primary goals. First, performing new research and development of enabling technologies in Design For Manufacturing and Assemble (DFMA). This R&D effort was centralized around the development of a DFMA analysis tool. The MO prototype system developed will analyze a unified product model represented in STEP against an

established manufacturing enterprise model. The manufacturing enterprise model captures an abstract representation of the manufacturing processes and resources supported by the enterprise. The enterprise model supports representation of reasoning logic that directly associates processes and resources of the enterprise to features of the product model. The analysis tool evaluates the reasoning logic and selects the appropriate processes and resources. Captured historical cost, yield and rework data from these processes or similar processes enable the system to estimate the cost, yield and rework rates for manufacturing the product under evaluation.

The second goal of the program was to "user harden" existing concurrent engineering technology. The following three tools developed under the DICE program were evaluated for inclusion in the MO system: 1) The Requirements Manager (RM), a system designed to manage product requirements, specifications and corporate policies to support concurrent engineering; 2) The Project Coordination Board (PCB) which provides support for the coordination of the product development activities in a cooperative environment; 3) The Rensselaer Object System for Engineering (ROSE) is an object data manager that has been developed for engineering applications and enhanced to support the DICE program.

2.4 Document Plan

The remainder of this document proceeds as follows: Section 3 details the planned activities of the DICE MO program and the methodology used in performing the research and the development of the prototype. Section 4 provides a discussion of the results of the program. The research performed and the software developed are presented. Section 5 presents some conclusions reached based on our research. Section 6 provides some recommended further direction of the results of the program. Section 7 provides a bibliography of the references used in creating this report. Section provides a list of acronyms and a glossary of key terms used throughout the report. The information required to represent manufacturing processes and resources is modeled in EXPRESS and is presented in Appendix I. An EXPRESS based information model for the PWB domain is presented in Appendix II. The user interface screens of the MO prototype are provided in Appendix III.

3 Methods, Assumptions and Procedures

This section is devoted to establishing the assumptions made, the procedures established, and the methods used during the course of the program.

3.1 Evaluation of Existing CE Technology

This section addresses "user hardening" of existing concurrent engineering technology by applying existing DICE tools to other technology areas, in this case Design for Manufacturing and Assembly (DFMA). The DICE tools proposed for use under the MO program included ROSE Database Management System (DBMS), Project Coordination Board (PCB), and the Requirements Manager (RM).

ROSE is an object-oriented database management system that has been developed for engineering applications and enhanced to support the DICE program. Raytheon planned on using ROSE to store and manage the data files for the manufacturing processes, operations, and steps, as well as, the various manufacturing analysis results.

The Project Coordination Board (PCB) is a system being developed to provide support for the coordination of the product development activities in a cooperative environment. Raytheon planned on using the PCB for monitoring of the overall product design cycle activities.

The Requirements Manager (RM) is a system designed to manage product requirements, specifications and corporate policies to support concurrent engineering. Within the MO program, Raytheon planned to use the RM for manufacturability/producibility guidelines.

Each of these tools listed above were evaluated for potential use in the DICE-MO system. See section 4.1 for a summary of the tools that are used in the MO system. Reference [2] provides the results of the analysis performed on the tools.

3.2 Collaborative DFMA Tool

The DICE-MO prototype system is designed to work as a collaborative DFMA Tool. The flexibility built into the system through its usage of EXPRESS based product models provides its generality. The foundation of the system is its ability to couple process and

resource selection directly to product features and attributes. The modeler is designed to work with any distinct product domain modeled in EXPRESS. For a given product domain, the modeler will provide the user with the ability to incorporate the EXPRESS entity definitions from that domain directly into the selection rules defined for manufacturing activities. When the product domain changes the set of entities available for use in the modeler will change to those of the EXPRESS model for the new product domain.

3.3 Prototype Development Methodology

The central concept behind object-oriented technology is of course that of the object. Briefly put, an object is a software unit comprised of information and operations that can be performed on that information. A central tenet of the concept is that applications should be built on top of models, with each model being used for more than one application. As data and operations are encapsulated within objects, applications become intertwined and begin to lose their separate identity. The focus of the computing effort goes from developing separate applications to refining and using shared models [Taylor90].

Analysis is primarily a process of discovery, through which the development team seeks to model and understand the real world. Design is primarily a process of invention and adaptation, during which the development team creates the abstractions and mechanisms necessary to meet the system's behavioral requirements as derived through analysis [Booch]. Throughout the analysis / design cycles strategic and tactical decisions are identified and made. The strategic decisions having sweeping architectural implications and the tactical decisions represent the essential details required to fulfill the envisioned system.

The DICE-MO prototype was developed using the object oriented paradigm. A continuous cycle of analysis, design and prototype was employed to accurately model the enterprise and to maximize system expectations. Figure 3-2 provides an illustration of the methodology employed while developing the DICE-MO prototype.

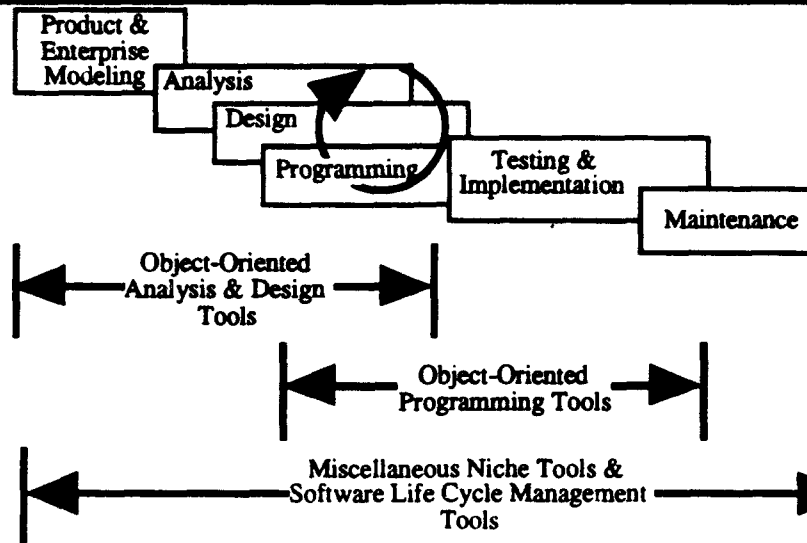


Figure 3-2. DICE-MO Prototype Development Methodology

The primary development platform used to design and develop the prototype was a Sun Sparc workstation. The primary source code language used for development was C++. State of the Art software development methods and tools were utilized extensively and are detailed in Table 3-1 below.

Table 3-1. Software Methods and Tools used in Prototype Development

Method / Tool Name	Development Phase	Description
EXPRESS information modeling	Product, Enterprise modeling	The EXPRESS information modeling language was used to model the information requirements of the prototype.
EDEX	Product, Enterprise modeling	EXPRESS extensions to the emacs text editor that provides for EXPRESS construct templating and syntax checking.
Coad / Yourdon notation	Analysis	OOA notation for modeling classes, objects, services.
Grady Booch notation	Design	OOD notation for modeling classes, objects, services. State transition diagrams and services' pseudo code are also featured
UIM/X	Analysis / Programming / Design	Graphical user interface builder. Generates Motif code in C and C++. Used for prototyping and final implementation of user interface.
Express2C++	Analysis / Programming / Design	Compiler that checks the syntax of EXPRESS and generated a C++ class hierarchy including access and update methods for the entities defined in the EXPRESS model.

STEPToolkit	Analysis / Programming / Design	Provides an EXPRESS-G diagram generator and editor. STEP Physical file input / output capabilities (SDAI).
Purify	Testing & Implementation	Memory error detection software.
Vista	Testing & Implementation	Monitors test cases and detects untested portions of source code.
Clearcase	Maintenance	Source Code Control

3.3.1 Product and Enterprise Information Modeling

The initial phase of prototype development consisted of modeling the information required by the product(s) that the MO system was targeted towards and the enterprise functions that were required to produce that product. In this case the initial product test case chosen was the Printed Wiring Board (PWB) design domain and the enterprise activities targeted were those in manufacturing required to fabricate and assemble PWBs.

EXPRESS was chosen as our information modeling language. This was based on several reasons including:

- EXPRESS is object-oriented and supports class hierarchy definition.
- EXPRESS is an International Standards Organization (ISO) standard for the specification of information models.
- The information modeling language used exclusively for development of the ISO STEP standard is EXPRESS.
- EXPRESS is computer interpretable and therefore can be used as input to C, C++ and SQL code generators.
- A graphical subset of the language exists called EXPRESS-G. The graphical nature of EXPRESS-G makes it a valuable tool for understanding and analyzing information models.

The information models generated for the PWB domain were based on work performed on an internal Raytheon project called RAPIDS (Raytheon's Automated Placement and Interconnect Design System). RAPIDS is used at Raytheon as a concurrent engineering station (or framework) for PWB design. The data set that RAPIDS supports is a superset of the data sets used by the various layout, analysis, and manufacturing systems in use at Raytheon for PWB design. The EXPRESS information model for the PWB domain was developed such that it supported the same information that the RAPIDS data model does.

For enterprise functions requiring modeling, a different approach was taken. A generic model was created that supported modeling manufacturing activities and resources. The generic model supported a very limited amount of specific attributes of any given activity or resource, but provides a powerful means of specifying the conditions under which a specific activity or resource would be used in manufacturing a product. These conditions are referred to as "selection rules". The Manufacturing Activity and Resource information models can be found in Appendix I.

3.3.2 Object-Oriented Analysis

An object oriented analysis (OOA) model was constructed in the Coad / Yourdon [COAD] notation that depicted the prototype's functional requirements. The model included specifications for class and objects, generalization-specialization and whole-part definitions of classes, and the services that classes would provide.

3.3.3 Object-Oriented Design

The object-oriented design (OOD) of the prototype was carried out by refining the OOA model with more specifics. Messages detailing object interaction were defined. State transition diagrams were presented providing a model of event sequences. High level pseudo code was provided to present detail of class services.

In addition to the OOD, prototype user interface screen images were provided. This was to provide the reader with a better feel of the system appearance and the user's methods for interacting with the system.

3.3.4 Object-Oriented Programming

The system is written in the C++ programming language. Two tools (express2c++ and UIM/X) were used extensively to generate code. Express2C++ generated a class library of access and update services for the entities defined in the EXPRESS information models. UIM/X is graphical user interface development tool that automatically generates the X-Window Motif library function calls necessary to generate the interface specified in the tool. These two tools enabled more time to be spent in analysis and design and less in implementation.

3.3.5 Testing and Implementation

The individual object class services and the user interface were tested using ad hoc data. The "purify" tool from Pure Software was used during these tests to ensure against memory corruptions and leaks. Once this unit testing was complete, the processes and resources used in PWB fabrication and assembly were modeled using the MO Modeler. Product data test cases were taken from existing PWB designs for PATRIOT ground system modules. The process selection, cost, yield, and rework analyses were then carried out and the results presented in the MO Advisor. To ensure proper test coverage, the "vista" test case validation tool from Veritas Software was used.

3.3.6 Maintenance

The information models, source code, and the test cases are controlled using the "Clearcase" source code control and configuration tool from Atria Software.

4 Results and Discussion

4.1 CE Technology Evaluation

The DICE tools that the MO program proposed to use are the ROSE database from STEP Tools Inc., ProductTrack Requirements Manager (RM) from CIMFLEX Teknowledge, and the Project Coordination Board (PCB) from Concurrent Engineering Research Center (CERC). For demonstration purposes, an interface was developed between Raytheon's Automated Placement and Interconnect Design System (RAPIDS) and the ROSE DB.

ROSE is an object-oriented database management system that has been developed for engineering applications and enhanced to support the DICE program. ROSE is provided as part of the STEP Programmer's Toolkit from STEP Tools, Inc. ROSE is a database which supports concurrency using a data model that allows the differences between two design versions to be computed as a delta file. The ROSE DB was evaluated and found to be suitable for the proposed task. Therefore, all data used by the MO prototype is stored and managed within ROSE.

The Requirements Manager (RM) is a software tool designed to manage product requirements and evaluate the compliance of product design data with requirements. The purpose of integrating the RM into the MO system was to provide the "top level" product development team insight into manufacturing requirements. It is common practice for a manufacturer to document manufacturability, or producibility guidelines which delineate standard manufacturing practices and acceptable design parameters. The purpose of these static guidelines is to communicate the capabilities of the manufacturing process to the product design community to ensure that new product designs are specified within manufacturing capabilities. The RM was evaluated as a stand-alone system. Raytheon believes that the system is capable of modeling and evaluating manufacturing guidelines. The plan was to implement a two way interface between MO and the RM using the RM API. Although planned, the RM API never supported two way communication, only an export function. Raytheon believes that a two way interface would be advantages to the RM.

The Project Coordination Board (PCB) provides support for the coordination of the product development activities in a cooperative environment. It provides common visibility and change notifications. The PCB was evaluated as a means to support the communication of the product/process development activities within the product design cycle. There was never plans for a direct interface between the MO software modules and the PCB applications. A simple model was entered into the PCB to evaluate its viability. The PCB proved too immature a software product to be of added value to the MO system. Figure 4-1 shows a high level view of the design cycle steps which we modeled in the PCB during the evaluation.

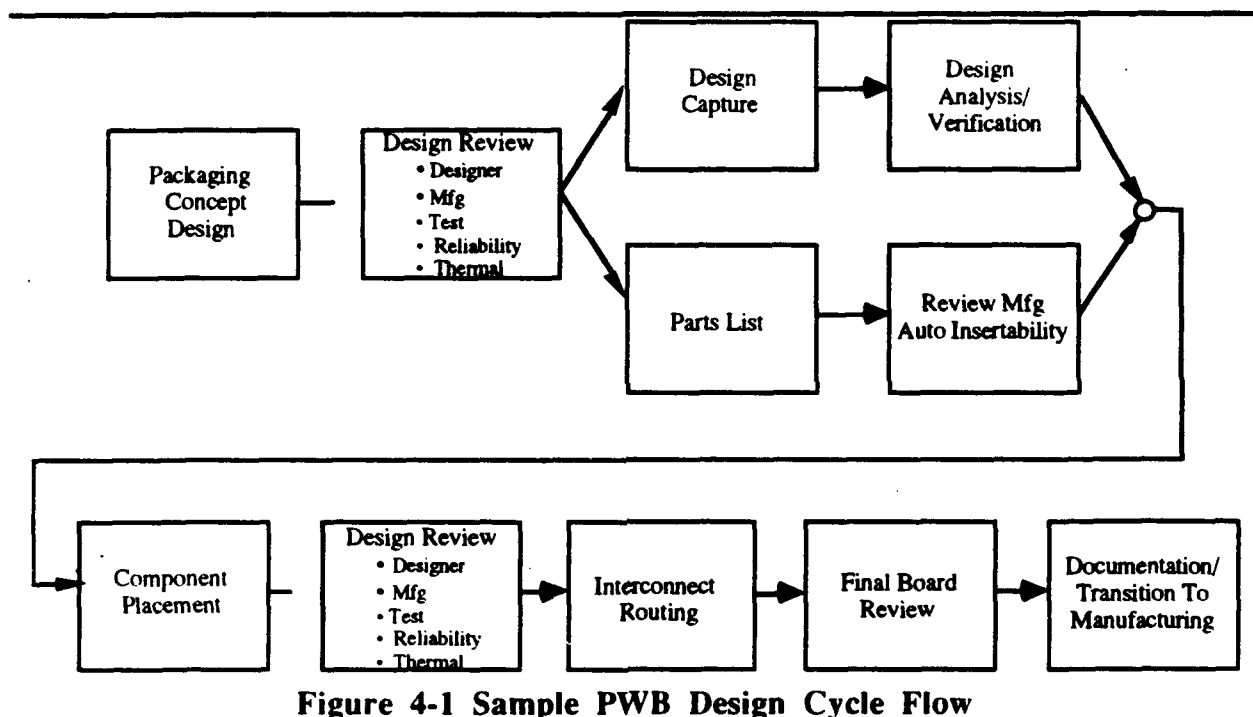


Figure 4-1 Sample PWB Design Cycle Flow

4.2 Information Modeling

Information models were developed for the representation of manufacturing activities (processes, operations and steps), manufacturing resources, and a sample product domain (Printed Wiring Board). Appendices I and II contain the information models developed for processes and resources and the PWB product domain respectively.

EXPRESS is an emerging International Standards Organization (ISO) language for the specification of information models. It was originally developed to enable a formal specification of the forthcoming ISO 10303 standard, familiarly known as STEP. The

language is also increasingly being used in many other contexts, for example in the mechanical, electronic and petro-chemical industries, as well as in other national and international standards efforts. EXPRESS-G is a graphical subset of the EXPRESS language. The graphical nature of EXPRESS-G makes it a valuable tool for understanding and analyzing information models. For these reasons, EXPRESS and EXPRESS-G were used to develop the information models.

4.3 Prototype System Architecture

MO is an X-Windows based tool. The application software is written in C and C++, the Motif user interface was developed using the UIM/X User Interface Management System, and all data is being stored in STEP physical files.

The STEP data file format standards were selected for MO to provide an early test of STEP, the emerging international standard for data exchange between automation systems. MO reads and writes STEP entity instances in these files through a C++ class library provided by STEP Programmer's Toolkit.

The MO core system is composed of three software modules, Manufacturing Analyzer, Manufacturing Advisor, and Process Modeler. The ROSE database from STEP Tools Inc., and the two way interface to the Raytheon Automated Placement and Interconnect Design System (RAPIDS) complete the software suite which constitute the MO system. Figure 4-2 illustrates the MO System Architecture.

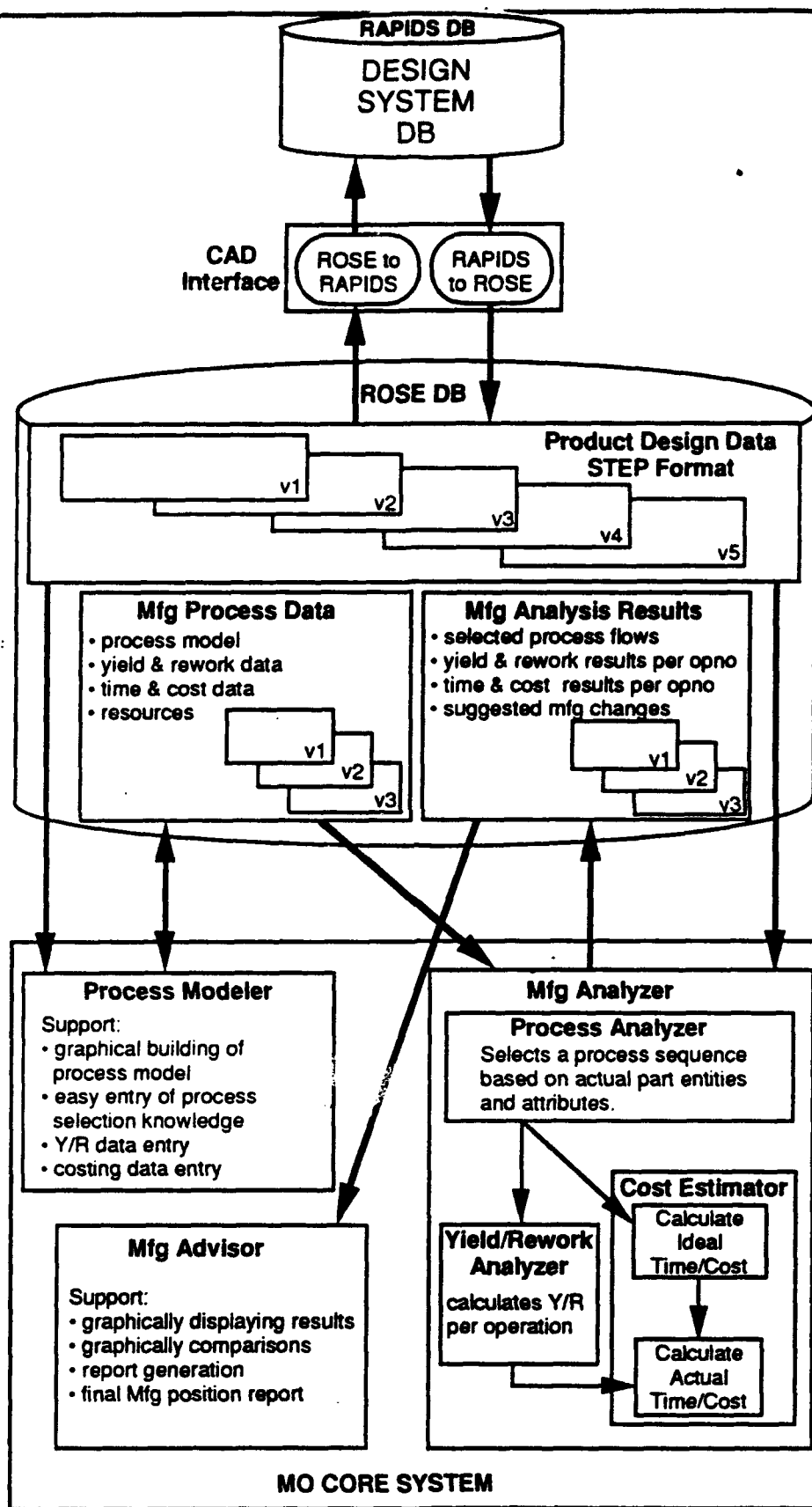


Figure 4-2 MO System Architecture

The Process Modeler enables the user to model processes and resources required to manufacture a product. Each process is modeled as a set of operations, where an operation is a unit of work performed on the product part. Each operation is modeled as a set of operational steps, where a step is an elemental unit of work within an operation. Yield and rework rates are defined for each operation. The output of the Process Modeler is a hierarchical tree structure of individual manufacturing activities which point at either process, operation, or step data. Figure 4-3 depicts a block diagram of the Process Modeler.

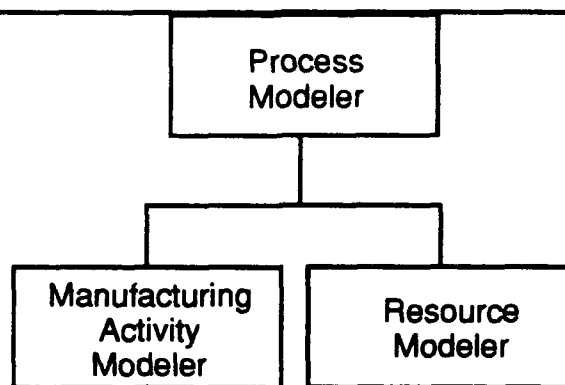


Figure 4-3 Process Modeler Block Diagram

The Manufacturing Analyzer provides the following three services: 1. Select the individual activities from the process model that are used to manufacture a particular product. 2. Analyze the processes, operations, and steps to estimate scrap and rework rates. 3. Analyze the resources attached to the selected processes, operations, and steps for cost. The analyzer results are composed of design feature entities from the product design database (STEP file) along with the selected manufacturing processes from the user specified process model. Figure 4-4 depicts a functional block diagram of the Manufacturing Analyzer.

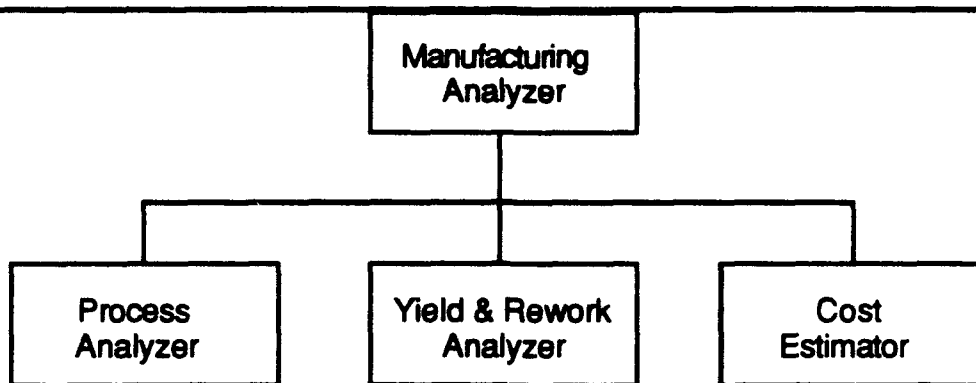


Figure 4-4 Manufacturing Analyzer Block Diagram

The Manufacturing Advisor provides the user with various methods to view the results produced from the analyzer. The results can be viewed graphically (i.e. line, bar, stacked bar and pie charts) or textually. The reporting capability allows the user to customize a detailed report which can be printed to the screen or to an ASCII file. MO allows the user to view one or more sets of analysis results at a time. By selecting multiple analysis runs to graphically display, the user can visually compare the analyses. Figure 4-5 shows a functional block diagram of the Manufacturing Advisor.

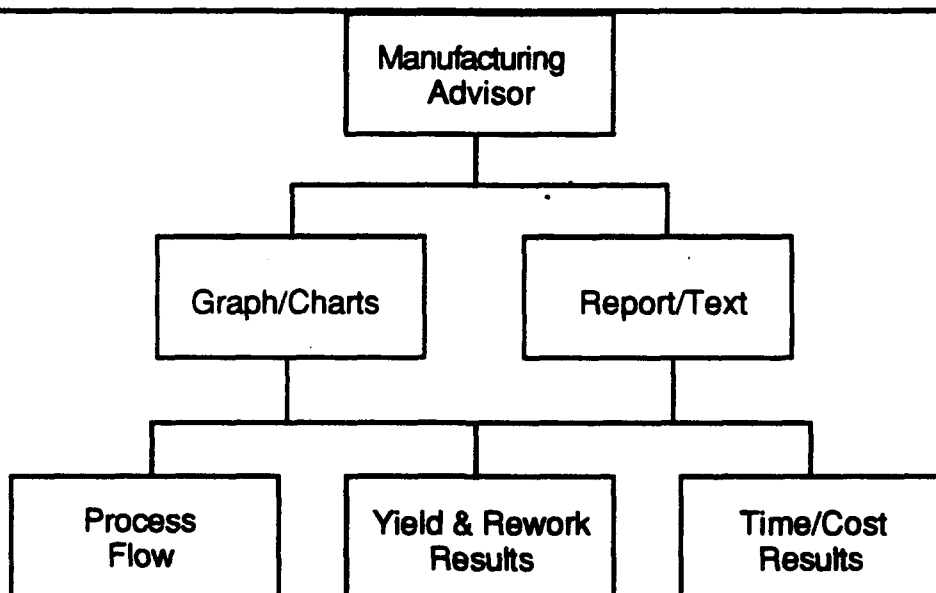


Figure 4-5 Manufacturing Advisor Block Diagram

4.4 Prototype System Components

4.4.1 Product (STEP) Models

The STEP data file format standards were selected for MO to provide an early test of STEP, the emerging international standard for data exchange between automation systems. MO reads and writes STEP entity instances in these files through a C++ class library provided by STEP Programmer's Toolkit. This class library is currently being updated to adhere to the ISO Part 22 SDAI (Standard Data Access Interface) specification.

At Raytheon, PWB product data is stored in the RAPIDS (Raytheon's Automated Placement and Interconnect Design System) database. Two interfaces were developed to support the transition of PWB product data to and from STEP physical files.

Generating the STEP physical file is facilitated by the *RAPIDS to STEP* interface which maps RAPIDS data items into instantiated STEP entities. An information model using the EXPRESS information modeling language was created based on the RAPIDS database. The EXPRESS information model was compiled using the STEP Tools *express2c++* compiler, which generated a STEP schema and a C++ class library. The class library consists of methods for creating and referencing persistent instances of the STEP entities which are stored in a ROSE database. The STEP schema is used by the STEP Tools *STEP filer* for reading and writing the STEP physical file.

The MO system uses the STEP data directly, as well as, for information exchange between the members of the product design team. For demonstration purposes, we will have the top level team using RAPIDS. This is not a requirement for using the MO system. The only requirement is that the top level team and the lower level teams be capable of creating, exchanging and using the STEP physical file.

The Manufacturing Team passes back a consolidated manufacturing position to the product design team. To aid in the generation of a consolidated position, conflict resolution and design merging must be supported. This is done using the STEP Toolkit from STEP Tools Inc. The *diff* tool reads two versions of a design and creates a delta file. The *difference report generator* reads the difference file and the original design, and presents each STEP

entity and its attributes with the original values and its change state clearly marked with an asterisks.

Once the conflicts of the Manufacturing team members have been resolved, design versions are merged using the STEP Tools *sed* tool. The *sed* tool reads the delta file created by the *diff* tool and updates the original design version. This updated version of the design will be transferred back to the top-level product team as the Manufacturing Team's consolidated position.

4.4.2 Process Models

The key to performing manufacturability analysis is to characterize the fabrication and assembly processes. In MO, this characterization is implemented as a manufacturing process representation and selection algorithm. Basing manufacturing cost analysis on a detailed description of the process provides visibility into the relationship between the design attributes and the manufacturing process. This allows the engineer to focus on manufacturing cost drivers and their causes. By characterizing the process in this manner, the manufacturing engineer is able to review the process which will be used to produce the product and be readily able to consider alternative manufacturing processes and their consequences.

Following this logic, it makes sense to capture the expert's process planning knowledge into a process selection model so that the relationship between the product entities and the process selected to fabricate the product is explicitly defined. This does not mean that there is a one to one relationship between the design entities and the process steps. In some areas, such as PWB, the design may be implemented using different technologies, each of which implies a certain process, such as surface mount versus through-hole technology. In other cases, there are multiple processes that can be used to produce the same entity. This is most prevalent in the metal fabrication (machining) area where often a number of processes (investment casting, milling) are capable of producing the part.

There were two development challenges: building a data schema to represent the manufacturing process such that it can be used for selection and costing, and building a selection logic algorithm that adequately represents the planning logic employed by expert process planning.

Normally in a manufacturing plant, the overall process for a given discipline is known and recorded in the form of a flowchart. This flowchart is a block diagram listing of each and every process within that discipline. The order of those operations is structured so that it is the default ordering of how products flow. If a process gets repeated, it generally shows up in each repeated point in the flow chart. These flowcharts usually employ a rudimentary decision logic scheme. As such it represents the available processes in a pick list fashion.

The logic representation method that Raytheon developed for this task is based on prior work in process selection. The model is hybrid of decision tree and rule based processing. The decision tree representation was selected because it allows the system to display the basic flow of the process in a presentation format similar to what the manufacturing engineers are used to with their flowcharts. The decision informs the user of the basic flow of the overall process while letting the user plan at various levels of abstraction. These levels include the process, an organized group of manufacturing operations sharing characteristics, the operation, a common unit of work that is performed on the part, and the operational step, which is an elemental unit of work within an operation. By defining the levels as we have, a hierarchical planning strategy is enabled. Using this schema, we can reason about alternative processes, plan the operation flow within the selected process, and then detail the individual steps of that operation, such as set-up and run time elements.

The reasoning process is guided by the representation of the tree structure which sets the initial search evaluation order, and the rule processing mechanisms. The reasoning logic is attached to individual activity nodes in the tree as selection rules. These rules are used to evaluate the node. Contained in the rules are references to the existence or value of product features. The purpose of the evaluation is to cause selection of the node. If a rule is evaluated as true then the search continues past that node to evaluate its lower levels. Once the tree traversal is complete, those activities that were selected, are evaluated for cost and yield drivers.

The system also has the ability to store alternative models of a particular process. This capability allows the process engineer the ability to explore alternative process approaches and plan process improvements. Figure 4-6 illustrates a sample assembly hierarchical tree for printed wiring boards.

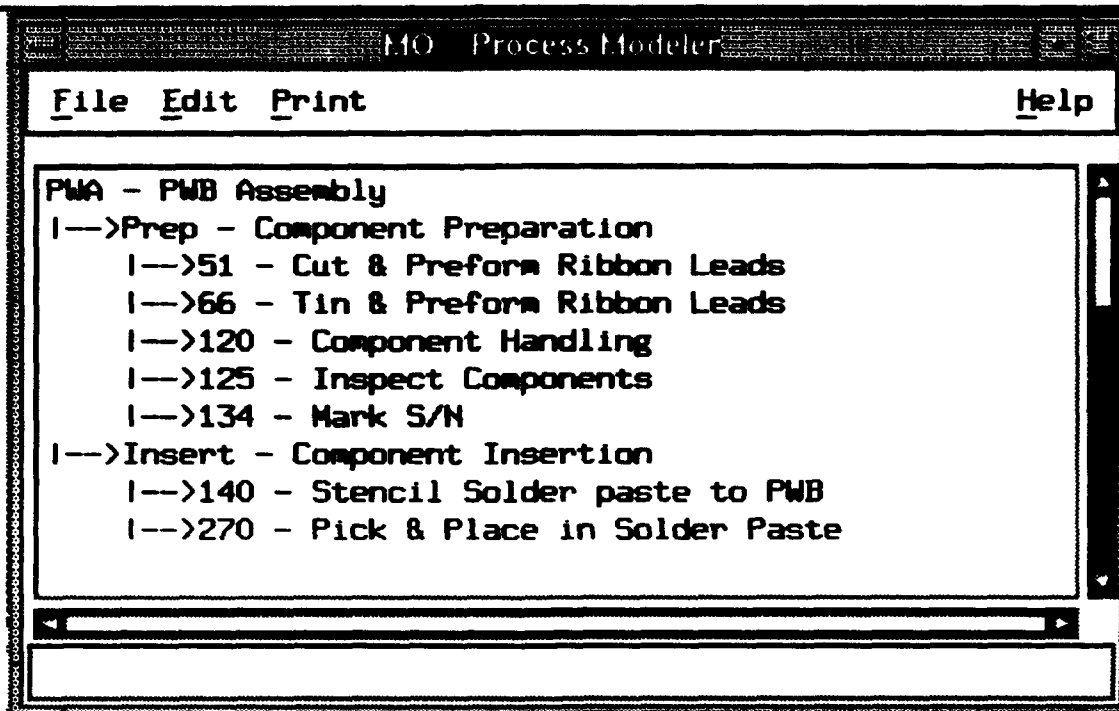


Figure 4-6 Printed Wiring Assembly Process Model

4.4.3 Manufacturing Analyzer

There are three capabilities provided in the Manufacturing Analyzer module: process analyzer, yield and rework analyzer, and cost estimator. The sub-sections to follow describe each capability.

4.4.3.1 Process Analyzer

The manufacturing process must be modeled to perform cost and yield analysis on a design. The MO process model supports a hierarchical tree based model of a manufacturing enterprise. Processes, operations and steps are defined for a manufacturing activity. Rules are defined which tie the product data to the processes, operations and steps. The selection rules, if satisfied, will trigger the selection of that process, operation or step.

An object-oriented methodology has been employed to implement the model. To represent processes, operations, and steps in the tree structure, a generic Manufacturing Activity class named "MfgSpec" was defined. The MfgSpec objects contain information that is common to processes, operations, and steps. Within each MfgSpec is a reference to an

"info" object. This info object contains the information specific to the type of manufacturing activity being modeled (i.e. process, operation, or step).

The Manufacturing Analyzer's selection methodology is done by traversing the process model in depth-first fashion. The logic at each manufacturing activity node will be evaluated to see if this is an applicable path to follow. The selected nodes are added to an analysis tree which is also modeled as a general purpose tree structure. After the entire process model has been evaluated and the applicable nodes identified, the analysis tree created during process selection is traversed in a post-order fashion so that the time and cost can be calculated.

4.4.3.2 Yield & Rework Analyzer

The yield and rework analyzer provides the capability to calculate yield and rework rates for the selected processes associated with a product design. This part of the analysis calculates the yield and/or rework rate on an operation level within the process. The rate is calculated based on the design entities influence on the operation. The yield and/or rework rate for each design entity/entities associated with an operation is calculated through the evaluation of a rule, which has a corresponding equation attached. If the rule evaluates to true, then the equation is calculated to provide the yield or rework rate. The rate equations may include references to the existence, value, or quantity of product design entities. An example yield rule and corresponding rate attached to an operation is as follows:

Yield Data:

Design Features Rule	Scrap Rate
aspect ratio < 5.0 & aspect ratio > 4.0	0.05000
aspect ratio <= 4.0 & aspect ratio > 2.0	0.02000

The total yield rate for an operation is calculated by using the weighted average of the constituent parts. The total rework rate for an operation is calculated by summing up the results of each rework occurrence.

4.4.3.3 Cost Estimator

The cost estimator calculates the recurring manufacturing cost for each activity in the process sequence. The following calculations are performed:

- Cost for each resource attached to a process, operation, and step are calculated for setup and run time utilization. The value for each is calculated through the evaluation of an equation which may include reference to the existence, value, or quantity of design entities in the product data. Each resource has an associated cost in terms of an appropriate measure. For example, a labor resource has an associated cost in terms of dollars per time unit.
- Estimated ideal cost for each process, operation, and step is calculated from labor standard values multiplied by the wage rate of the labor grade or bid code of the resource(s) performing the operation, and the production efficiency value for that operation.
- Rework operations are calculated based on the rework rate determined by the yield and rework analyzer multiplied by labor standards of the resources for the rework condition. The labor grade wage rates and production efficiencies are then applied.
- For each operation, the estimated actual cost is calculated by multiplying the estimated ideal cost by the number of units processed, including both good and scrapped units. The number of units processed by each operation are calculated from the value of the required good units at the subsequent operation divided by the yield at the operation under evaluation. For example, if the desired production quantity is 100 boards and operation 1 has a scrap rate of 5%, then the quantity of required units for operation 1 is 105.
- The total estimated ideal cost and total estimated actual cost for each sequence of processes are calculated by rolling up the individual cost of steps into operations, and operations into processes. The estimated actual cost for a good unit is calculated by dividing the total estimated actual cost for the process by the number of good units produced.

4.4.4 Manufacturing Advisor

The manufacturing advisor provides the capability to view the results produced by each activity participating in an analysis. The advisor includes the following capabilities:

- A mechanism for selecting one or more manufacturing analyzer runs for comparing and/or displaying the results.
- Graphical capabilities (i.e., line, bar, stacked bar and pie charts) for comparing and displaying the process, yield, rework, or cost versus a processes or operations for one or more manufacturing analyzer runs.
- A reporting capability which prints analyzer results to the screen or file for one or more runs including process sequence, yield and rework, and cost.

Provided below in figure 4-7 is a sample of the type of graphical display the user could see for a yield versus process comparison graph.

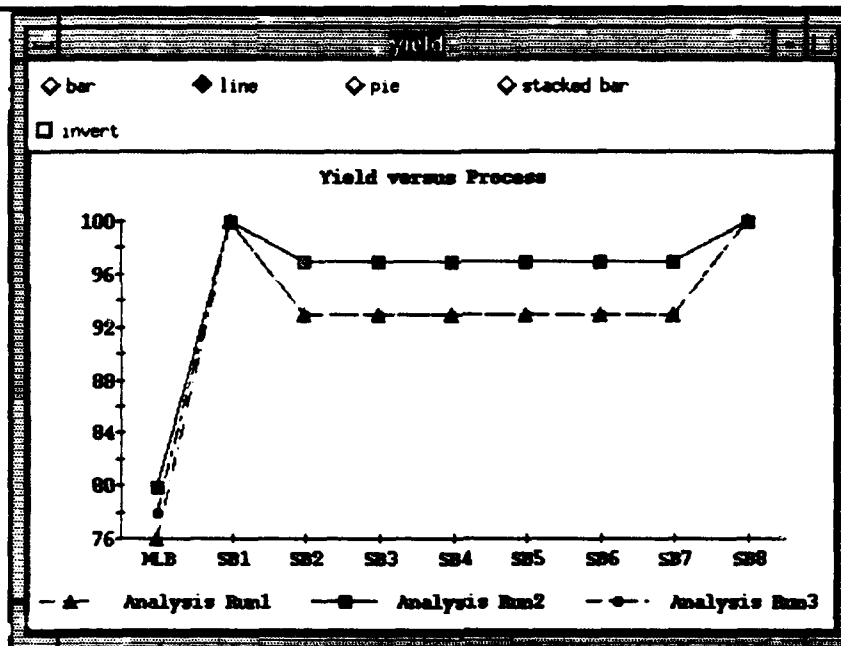


Figure 4-7 Sample Yield versus Process Comparison Graph

4.4.6 Process Modeler

The process modeler provides the capability to model the selection logic of the manufacturing process. The process model used in the MO system is designed as a hierarchical planning system. The hierarchical planning system is developed as a general purpose tree structure. The hierarchical tree consist of Manufacturing Activity nodes. Each Manufacturing Activity node consists of the following:

- Reasoning Logic - If these rules are satisfied, then the activity node is included in the total process analysis model.
- Manufacturing Data - There are three type of manufacturing data supported in the MO hierarchy model. The three data types are processes, operations, and steps. The data will be modeled by linking manufacturing processes to operations, and operations to steps. Each operation is annotated with its associated yield and rework rates.
- Resources - At each process, operation, or step node there is a list of resources attached. A resource is any facility, person, equipment, or consumable material used in the manufacturing process.
- Ordering - The children of a Manufacturing Activity node are defined with an imposed ordering of a concurrent or sequential flow when building the model.

The MO system allows the manufacturing specialists to capture and maintain multiple copies of process models through a set of utilities. The utilities provide the model developer with the tools necessary to graphically build and view the process logic tree, reasoning logic, yield/rework, resources, and labor standards. Through the use of these utilities, the process team has the ability to modify the process model data, to explore alternative process approaches and plan process improvements, and then analyze the effects of these changes on the product cost. The user interface consists of pull down menus and pop up forms to allow adding, copying, moving, deleting, editing, and printing of the processes in the hierarchical tree. Pictured below in figure 4-8 is the main user interface window for the Process Modeler with a sample process model displayed in a list view.

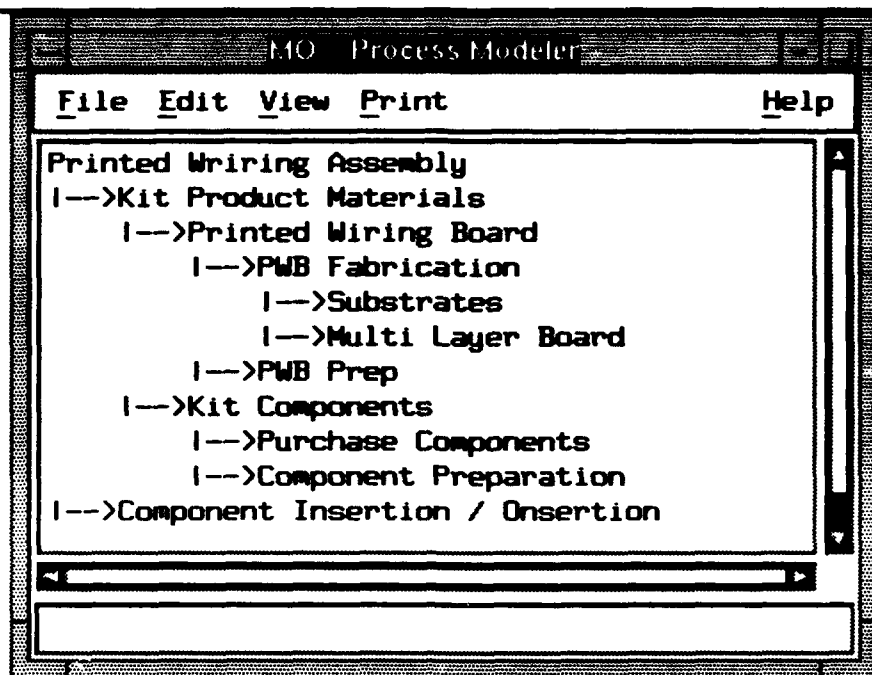


Figure 4-8 Process Modeler User Interface Window

4.4.7 RAPIDS

RAPIDS is Raytheon's conceptual design and analysis workstation for Printed Wiring Boards (PWB). RAPIDS supports component placement and placement density analysis, as well as a number of other analysis functions, including automatic component insertion checking. Interfaces between RAPIDS and the PWB analysis tools for the following criteria are also provided as part of the RAPIDS tool suite:

- Manufacturing
- Post Layout Effects
- Reliability
- Thermal

At Raytheon, RAPIDS is used for conceptual design and analysis of PWB's. RAPIDS serves in the same capacity at Raytheon that many commercial CAD systems (e.g. Mentor Board Station, Racal-Redac Visula, Cadence, etc.) are used in at other companies. RAPIDS provides an Application Programmatic Interface (API) with its database. This enables RAPIDS to be easily interfaced with other systems and standards. Using RAPIDS in the MO system is inline with Raytheon methodologies, but does not exclude interfacing MO with commercially available CAD systems in the future. The key to interfacing MO

with a large base of CAD systems is the utilization of the STEP standard by the commercial CAD industry.

4.5 Test Cases

4.5.1 PWB Fabrication and Assembly

The Printed Wiring Board (PWB) domain was used as the initial test vehicle for DICE-MO. The information requirements of the PWB domain were captured in an EXPRESS information model. Instantiations of the model were created through an interface to Raytheon's Automated Placement and Interconnect Design System (RAPIDS). Modules from the PATRIOT ground base system were translated into STEP product models. The Raytheon Andover PWB fabrication and assembly processes and the resources those processes utilize were modeled using the DICE-MO Modeler. Process flow, cost and yield analyses were performed on the PATRIOT modules by the DICE-MO Analyzer, and the results were accessed through the Advisor.

4.5.1.1 PWB Assembly Model

Provided below is the printed wiring board assembly hierarchical tree model, and following are the MO Process Modeler screens displaying various parts of this model.

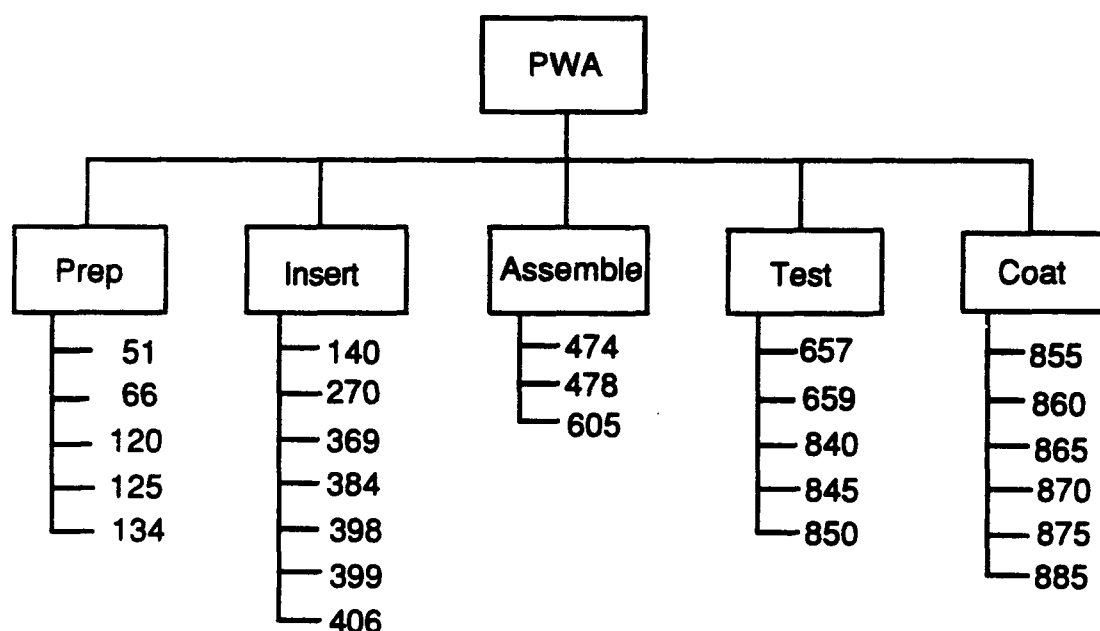


Figure 4-9 Printed Wiring Board Assembly Tree Model

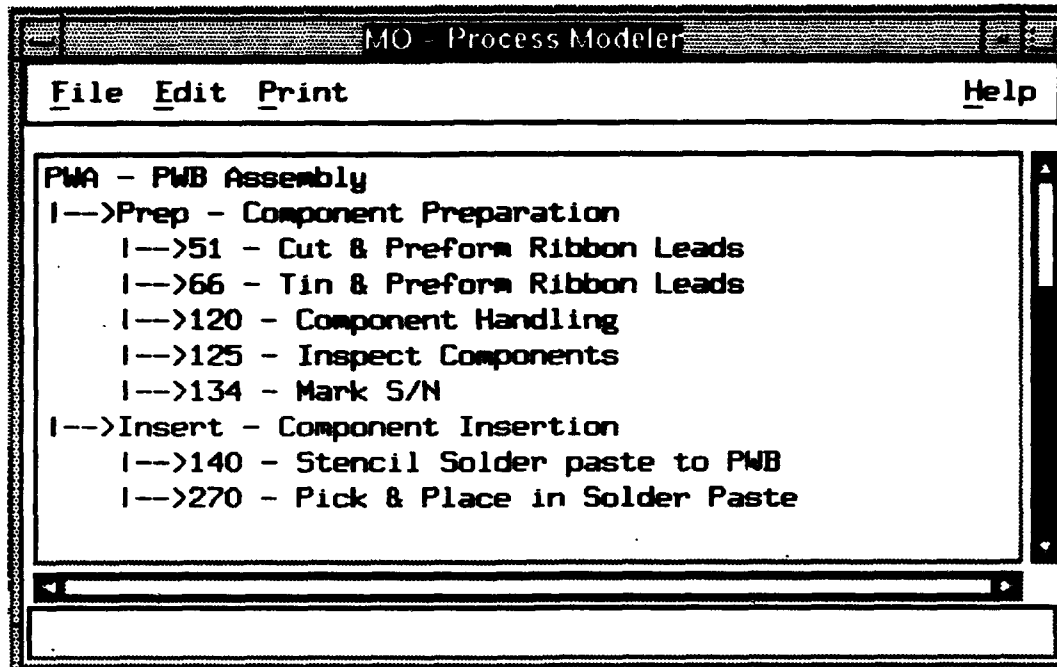


Figure 4-10 Printed Wiring Board Assembly Flow

The screenshot shows a dialog box titled "Manufacturing Activity Specification". It contains the following fields and controls:

- Parent :
- Name :
- Description :
- Activity :
 - ☒ Process
 - ☐ Operation
 - ☐ Setup Step
 - ☐ Action Step
- Child Ordering :
 - ☒ Serial
 - ☐ Concurrent
- Rules ...
- Resources ...
- Yield ...
- Rework ...
- OK
- Cancel
- Help

Figure 4-11 Process PWA Activity Specification

Selection Rules

Process Name : PWA

substrate_block_rec.technology="PWB"

New Edit Delete

OK Cancel Help

Figure 4-12 Process PWA Selection Rules

Manufacturing Activity Specification

Parent : Insert

Name : 270

Description : Pick & Place

Activity :

- ◇ Process
- ◆ Operation
- ◇ Setup Step
- ◇ Action Step

OK

Selection Rules

Process Name : 270

xref_rec.auto_insert="AUTOMATIC"

New Edit Delete

OK Cancel Help

Figure 4-13 Operation 270 Activity Specification & Selection Rules

4.5.1.2 PWB Fabrication Model

Provided below is the printed wiring board fabrication hierarchical tree model, and following it are the MO Process Modeler screens displaying various parts of this model.

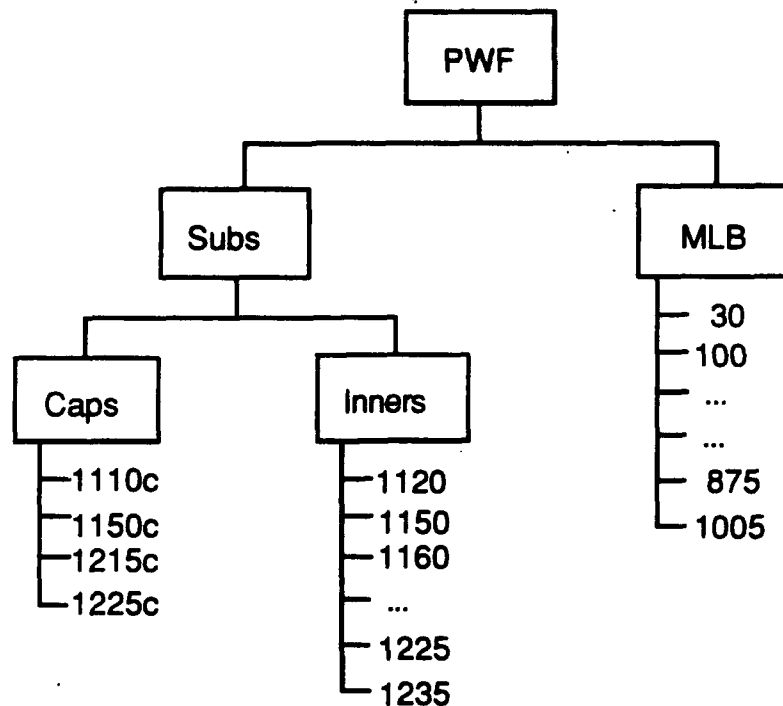


Figure 4-14 Printed Wiring Board Fabrication Tree Model

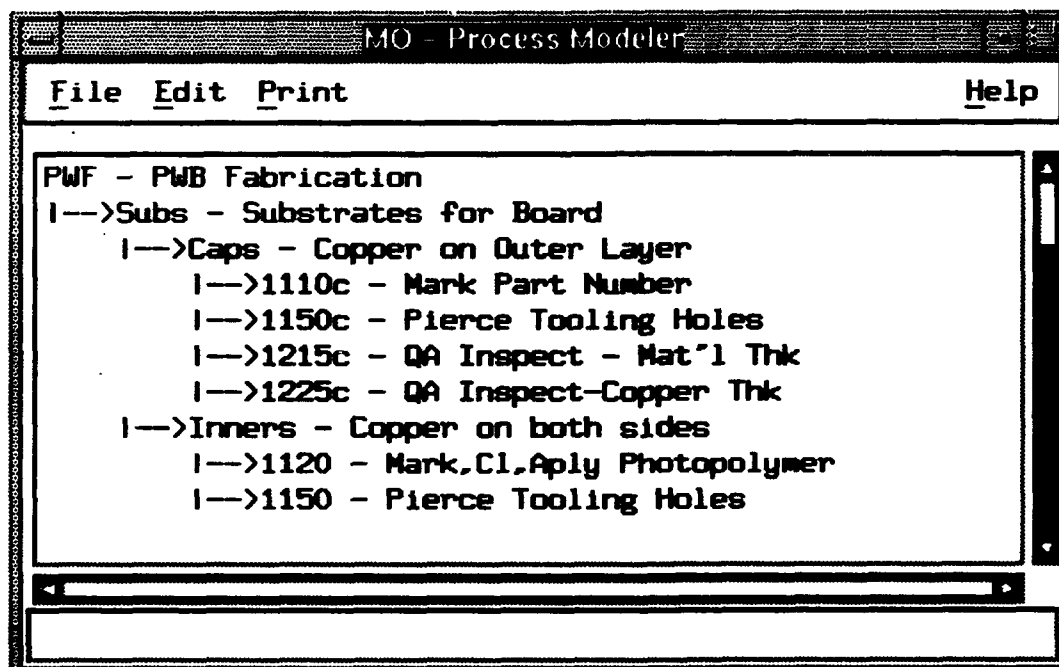


Figure 4-15 Printed Wiring Board Fabrication Flow

The screenshot shows the 'Manufacturing Activity Specification' dialog box. The 'Parent' field is empty. The 'Name' field contains 'PWF'. The 'Description' field contains 'PWB Fabrication'. The 'Activity' section has four radio buttons: 'Process' (selected), 'Operation', 'Setup Step', and 'Action Step'. The 'Selection Rules' dialog box is open over the main dialog. It has a 'Process Name' field containing 'PWF'. Below it is a text area with two rules: 'substrate_block_rec.technology="PWB"' and 'substrate_block_rec.technology="SURF"'. To the right of the text area are 'New', 'Edit', and 'Delete' buttons. At the bottom of the 'Selection Rules' dialog are 'OK', 'Cancel', and 'Help' buttons. The main dialog also has an 'OK' button at the bottom left.

Figure 4-16 Process PWF Activity Specification & Selection Rules

The screenshot shows the 'Manufacturing Activity Specification' dialog box. The 'Parent' field is empty. The 'Name' field contains '130'. The 'Description' field contains 'Lamination'. The 'Activity' section has four radio buttons: 'Process', 'Operation' (selected), 'Setup Step', and 'Action Step'. The 'Yield Specification' dialog box is open over the main dialog. It has an 'Operation Name' field containing '130'. Below it is a text area with five rules: 'substrate_block_rec.layers<=14 & substr', 'substrate_block_rec.layers<=11 & substr', 'substrate_block_rec.layers<=9 & substra', 'substrate_block_rec.layers<=7 & substra', and 'substrate_block_rec.layers=5'. To the right of the text area are 'New', 'Edit', and 'Delete' buttons. Below the text area is a 'Yield Rate' field containing '0.060000' and an 'Edit' button. At the bottom of the 'Yield Specification' dialog are 'OK', 'Cancel', and 'Help' buttons. The main dialog also has an 'OK' button at the bottom left.

Figure 4-17 Operation 130 Activity & Yield Specifications

4.5.1.3 PWA Analysis Report

MANUFACTURING PROCESS FLOW

<u>Name</u>	<u>Description</u>	<u>INDIVIDUAL TIME</u>		<u>TOTAL TIME</u>	
		<u>Ideal(hr)</u>	<u>Actual(hr)</u>	<u>Ideal(hr)</u>	<u>Actual(hr)</u>
PWA	PWB Assembly	0.000	0.00	1.77	1.78
Prep	Component Preparation	0.000	0.00	0.154	0.15
51	Cut & Preform Ribbon Leads	0.000	0.00	0.000	0.00
66	Tin & Preform Ribbon Leads	0.008	0.01	0.008	0.01
120	Component Handling	0.007	0.01	0.007	0.01
125	Inspect Components	0.034	0.03	0.034	0.03
134	Mark S/N	0.104	0.10	0.104	0.10
Insert	Component Insertion	0.000	0.00	1.055	1.06
140	Stencil Solder paste to PWB	0.000	0.00	0.000	0.00
270	Pick & Place in Solder Paste	0.000	0.00	0.000	0.00
369	Reflow Solder	0.343	0.34	0.343	0.34
384	Clean & Put in Fixture	0.092	0.09	0.092	0.09
398	Touch-Up Solder	0.571	0.57	0.571	0.57
399	Clean	0.031	0.03	0.031	0.03
406	Inspect	0.019	0.02	0.019	0.02
Assemble	Mechanical Assembly	0.000	0.00	0.048	0.05
474	Assemble & Reflow	0.000	0.00	0.000	0.00
478	Clean	0.031	0.03	0.031	0.03
605	Final Inspect	0.018	0.02	0.018	0.02
Test	Board Testing	0.000	0.00	0.208	0.21
657	Thermal Shock	0.000	0.00	0.000	0.00
659	In-Circuit Test	0.047	0.05	0.047	0.05
840	Hot Solvent Wash	0.031	0.03	0.031	0.03
845	Water Wash & Bake	0.007	0.01	0.007	0.01
850	Omega Test	0.124	0.12	0.124	0.12
Coat	Board Coating	0.000	0.00	0.310	0.31
855	Mask	0.000	0.00	0.000	0.00
860	Attach Labels	0.124	0.12	0.124	0.12
865	Bake	0.007	0.01	0.007	0.01
870	Spray Coat (Circuit Side)	0.119	0.12	0.119	0.12
875	Demask	0.018	0.02	0.018	0.02
885	Inspect	0.043	0.04	0.043	0.04

YIELD AND REWORK BREAKDOWN

<u>Name</u>	<u>Description</u>	<u>Yield</u>	<u>Rework(\$)</u>	<u>%Rework</u>	<u>ProdQTY</u>
PWA	PWB Assembly	1.00	0.00	0.000	100
Prep	Component Preparation	1.00	0.00	0.000	100
51	Cut & Preform Ribbon Leads	1.00	0.00	0.000	100
66	Tin & Preform Ribbon Leads	1.00	0.00	0.000	100
120	Component Handling	1.00	0.00	0.000	100
125	Inspect Components	1.00	0.00	0.000	100
134	Mark S/N	1.00	0.00	0.000	100
Insert	Component Insertion	1.00	0.00	0.000	100

140	Stencil Solder paste to PWB	1.00	0.00	0.000	100
270	Pick & Place in Solder Paste	1.00	0.00	0.000	100
369	Reflow Solder	1.00	0.00	0.000	100
384	Clean & Put in Fixture	1.00	0.00	0.000	100
398	Touch-Up Solder	1.00	0.00	0.000	100
399	Clean	1.00	0.00	0.000	100
406	Inspect	1.00	0.00	0.000	100
Assemble	Mechanical Assembly	1.00	0.00	0.000	100
474	Assemble & Reflow	1.00	0.00	0.000	100
478	Clean	1.00	0.00	0.000	100
605	Final Inspect	1.00	0.00	0.000	100
Test	Board Testing	1.00	0.00	0.000	100
657	Thermal Shock	1.00	0.00	0.000	100
659	In-Circuit Test	1.00	0.00	0.000	100
840	Hot Solvent Wash	1.00	0.00	0.000	100
845	Water Wash & Bake	1.00	0.00	0.000	100
850	Omega Test	1.00	0.00	0.000	100
Coat	Board Coating	1.00	0.00	0.000	100
855	Mask	1.00	0.00	0.000	100
860	Attach Labels	1.00	0.00	0.000	100
865	Bake	1.00	0.00	0.000	100
870	Spray Coat (Circuit Side)	1.00	0.00	0.000	100
875	Demask	1.00	0.00	0.000	100
885	Inspect	1.00	0.00	0.000	100

COSTING BREAKDOWN

<u>Name</u>	<u>Description</u>	<u>INDIVIDUAL COST</u>		<u>TOTAL COST</u>	
		<u>Ideal(\$)</u>	<u>Actual(\$)</u>	<u>Ideal(\$)</u>	<u>Actual(\$)</u>
PWA	PWB Assembly	0.000	0.00	20.014	20.01
Prep	Component Preparation	0.000	0.00	1.683	1.68
51	Cut & Preform Ribbon Leads	0.000	0.00	0.000	0.00
66	Tin & Preform Ribbon Leads	0.088	0.09	0.088	0.09
120	Component Handling	0.077	0.08	0.077	0.08
125	Inspect Components	0.389	0.39	0.389	0.39
134	Mark S/N	1.129	1.13	1.129	1.13
Insert	Component Insertion	0.000	0.00	11.441	11.44
140	Stencil Solder paste to PWB	0.000	0.00	0.000	0.00
270	Pick & Place in Solder Paste	0.000	0.00	0.000	0.00
369	Reflow Solder	3.714	3.71	3.714	3.71
384	Clean & Put in Fixture	0.996	1.00	0.996	1.00
398	Touch-Up Solder	6.188	6.19	6.188	6.19
399	Clean	0.331	0.33	0.331	0.33
406	Inspect	0.212	0.21	0.212	0.21
Assemble	Mechanical Assembly	0.000	0.00	0.531	0.53
474	Assemble & Reflow	0.000	0.00	0.000	0.00
478	Clean	0.331	0.33	0.331	0.33
605	Final Inspect	0.199	0.20	0.199	0.20
Test	Board Testing	0.000	0.00	2.632	2.63
657	Thermal Shock	0.000	0.00	0.000	0.00
659	In-Circuit Test	0.540	0.54	0.540	0.54
840	Hot Solvent Wash	0.331	0.33	0.331	0.33
845	Water Wash & Bake	0.070	0.07	0.070	0.07

850	Omega Test	1.691	1.69	1.691	1.69
Coat	Board Coating	0.000	0.00	3.727	3.73
855	Mask	0.000	0.00	0.000	0.00
860	Attach Labels	1.691	1.69	1.691	1.69
865	Bake	0.070	0.07	0.070	0.07
870	Spray Coat (Circuit Side)	1.287	1.29	1.287	1.29
875	Demask	0.190	0.19	0.190	0.19
885	Inspect	0.488	0.49	0.488	0.49

4.5.1.4 PWF Analysis Report

MANUFACTURING PROCESS FLOW

Name	Description	INDIVIDUAL TIME		TOTAL TIME	
		Ideal(hr)	Actual(hr)	Ideal(hr)	Actual(hr)
PWF	PWB Fabrication	0.000	0.00	20.806	20.84
Subs	Substrates for Board	0.000	0.00	1.660	1.66
Caps	Copper on Outer Layer	0.000	0.00	0.621	0.62
1110c	Mark Part Number	0.391	0.39	0.391	0.39
1150c	Pierce Tooling Holes	0.018	0.02	0.018	0.02
1215c	QA Inspect - Mat'l Thk	0.173	0.17	0.173	0.17
1225c	QA Inspect-Copper Thk	0.039	0.04	0.039	0.04
Inners	Copper on both sides	0.000	0.00	1.040	1.04
1120	Mark,Cl.Aply Photopolymer	0.489	0.49	0.489	0.49
1150	Pierce Tooling Holes	0.018	0.02	0.018	0.02
1160	Expose Photopolymer	0.068	0.07	0.068	0.07
1170	Develop Photopolymer	0.050	0.05	0.050	0.05
1190	Etch Copper	0.046	0.05	0.046	0.05
1200	Strip Photopolymer	0.013	0.01	0.013	0.01
1215	QA Inspect - Mat'l Thk	0.173	0.17	0.173	0.17
1225	QA Inspect-Copper Thk	0.039	0.04	0.039	0.04
1235	QA Inspect Scanning	0.145	0.14	0.145	0.14
MLB	Build Multi-Layer Board	0.000	0.00	19.145	19.18
30	Oxide Treatment	0.133	0.14	0.133	0.14
100	Bake Panels	0.032	0.03	0.032	0.03
110	Lay up for lamination	0.342	0.37	0.342	0.37
130	Laminate	0.044	0.05	0.044	0.05
140	Lamination Teardown	0.323	0.32	0.323	0.32
160	Stress Relieve	0.006	0.01	0.006	0.01
170	Rout Excess Material	0.056	0.06	0.056	0.06
220	Drill	15.366	15.37	15.366	15.37
250	Pressure Blast	0.069	0.07	0.069	0.07
280	Bake	0.000	0.00	0.000	0.00
290	Plasma Desmear	0.239	0.24	0.239	0.24
300	Glass Etch	0.072	0.07	0.072	0.07
440	Electroless CU Deposition	0.048	0.05	0.048	0.05
460	Electrostrike Copper	0.042	0.04	0.042	0.04
480	Scrub	0.021	0.02	0.021	0.02
510	Apply Photopolymer	0.059	0.06	0.059	0.06
520	Expose Photopolymer	0.068	0.07	0.068	0.07
530	Develop Photopolymer	0.050	0.05	0.050	0.05
560	Copper/Tin Lead Plate	0.116	0.12	0.116	0.12
580	Mark Control Number	0.039	0.04	0.039	0.04

605	QA Inspect-Tin/Lead	0.016	0.02	0.016	0.02
620	Strip Photopolymer	0.023	0.02	0.023	0.02
680	Etch Copper	0.046	0.05	0.046	0.05
710	Bake	0.006	0.01	0.006	0.01
720	Reflow	0.034	0.03	0.034	0.03
770	Stencil	0.073	0.07	0.073	0.07
775	QA Inspect Scanning	0.054	0.05	0.054	0.05
780	Rout Boards and Coupons	0.094	0.09	0.094	0.09
795	QA Inspect	0.035	0.03	0.035	0.03
815	Coupon Prep	0.090	0.09	0.090	0.09
845	QA Inspect Final	1.465	1.47	1.465	1.47
875	QA Electrical Test	0.024	0.02	0.024	0.02
1005	MIR Inspect	0.060	0.06	0.060	0.06

YIELD AND REWORK BREAKDOWN

<u>Name</u>	<u>Description</u>	<u>Yield</u>	<u>Rework(\$)</u>	<u>%Rework</u>	<u>ProdQTY</u>
PWF	PWB Fabrication	0.94	0.00	0.000	100
Subs	Substrates for Board	1.00	0.00	0.000	100
Caps	Copper on Outer Layer	1.00	0.00	0.000	100
1110c	Mark Part Number	1.00	0.00	0.000	100
1150c	Pierce Tooling Holes	1.00	0.00	0.000	100
1215c	QA Inspect - Mat'l Thk	1.00	0.00	0.000	100
1225c	QA Inspect-Copper Thk	1.00	0.00	0.000	100
Inners	Copper on both sides	1.00	0.00	0.000	100
1120	Mark,Cl,Aply Photopolymer	1.00	0.00	0.000	100
1150	Pierce Tooling Holes	1.00	0.00	0.000	100
1160	Expose Photopolymer	1.00	0.00	0.000	100
1170	Develop Photopolymer	1.00	0.00	0.000	100
1190	Etch Copper	1.00	0.00	0.000	100
1200	Strip Photopolymer	1.00	0.00	0.000	100
1215	QA Inspect - Mat'l Thk	1.00	0.00	0.000	100
1225	QA Inspect-Copper Thk	1.00	0.00	0.000	100
1235	QA Inspect Scanning	1.00	0.00	0.000	100
MLB	Build Multi-Layer Board	0.94	0.00	0.000	100
30	Oxide Treatment	1.00	0.00	0.000	107
100	Bake Panels	1.00	0.00	0.000	107
110	Lay up for lamination	1.00	0.00	0.000	107
130	Laminate	0.94	0.00	0.000	107
140	Lamination Teardown	1.00	0.00	0.000	100
160	Stress Relieve	1.00	0.00	0.000	100
170	Rout Excess Material	1.00	0.00	0.000	100
220	Drill	1.00	0.00	0.000	100
250	Pressure Blast	1.00	0.00	0.000	100
280	Bake	1.00	0.00	0.000	100
290	Plasma Desmear	1.00	0.00	0.000	100
300	Glass Etch	1.00	0.00	0.000	100
440	Electroless CU Deposition	1.00	0.00	0.000	100
460	Electrostrike Copper	1.00	0.00	0.000	100
480	Scrub	1.00	0.00	0.000	100
510	Apply Photopolymer	1.00	0.00	0.000	100
520	Expose Photopolymer	1.00	0.00	0.000	100
530	Develop Photopolymer	1.00	0.00	0.000	100

560	Copper/Tin Lead Plate	1.00	0.00	0.000	100
580	Mark Control Number	1.00	0.00	0.000	100
605	QA Inspect-Tin/Lead	1.00	0.00	0.000	100
620	Strip Photopolymer	1.00	0.00	0.000	100
680	Etch Copper	1.00	0.00	0.000	100
710	Bake	1.00	0.00	0.000	100
720	Reflow	1.00	0.00	0.000	100
770	Stencil	1.00	0.00	0.000	100
775	QA Inspect Scanning	1.00	0.00	0.000	100
780	Rout Boards and Coupons	1.00	0.00	0.000	100
795	QA Inspect	1.00	0.00	0.000	100
815	Coupon Prep	1.00	0.00	0.000	100
845	QA Inspect Final	1.00	0.00	0.000	100
875	QA Electrical Test	1.00	0.00	0.000	100
1005	MIR Inspect	1.00	0.00	0.000	100

COSTING BREAKDOWN

Name	Description	INDIVIDUAL COST		TOTAL COST	
		Ideal(\$)	Actual(\$)	Ideal(\$)	Actual(\$)
PWF	PWB Fabrication	0.000	0.00	252.042	252.49
Subs	Substrates for Board	0.000	0.00	19.387	19.39
Caps	Copper on Outer Layer	0.000	0.00	7.269	7.27
1110c	Mark Part Number	4.625	4.63	4.625	4.63
1150c	Pierce Tooling Holes	0.234	0.23	0.234	0.23
1215c	QA Inspect - Mat'l Thk	1.970	1.97	1.970	1.97
1225c	QA Inspect-Copper Thk	0.440	0.44	0.440	0.44
Inners	Copper on both sides	0.000	0.00	12.118	12.12
1120	Mark,Cl.Aply Photopolymer	5.738	5.74	5.738	5.74
1150	Pierce Tooling Holes	0.234	0.23	0.234	0.23
1160	Expose Photopolymer	0.775	0.78	0.775	0.78
1170	Develop Photopolymer	0.567	0.57	0.567	0.57
1190	Etch Copper	0.584	0.58	0.584	0.58
1200	Strip Photopolymer	0.159	0.16	0.159	0.16
1215	QA Inspect - Mat'l Thk	1.970	1.97	1.970	1.97
1225	QA Inspect-Copper Thk	0.440	0.44	0.440	0.44
1235	QA Inspect Scanning	1.650	1.65	1.650	1.65
MLB	Build Multi-Layer Board	0.000	0.00	232.655	233.11
30	Oxide Treatment	1.689	1.81	1.689	1.81
100	Bake Panels	0.369	0.39	0.369	0.39
110	Lay up for lamination	3.892	4.16	3.892	4.16
130	Laminate	0.503	0.54	0.503	0.54
140	Lamination Teardown	3.678	3.68	3.678	3.68
160	Stress Relieve	0.064	0.06	0.064	0.06
170	Rout Excess Material	0.715	0.72	0.715	0.72
220	Drill	188.385	188.39	188.385	188.39
250	Pressure Blast	0.876	0.88	0.876	0.88
280	Bake	0.000	0.00	0.000	0.00
290	Plasma Desmear	3.031	3.03	3.031	3.03
300	Glass Etch	0.909	0.91	0.909	0.91
440	Electroless CU Deposition	0.604	0.60	0.604	0.60
460	Electrostrike Copper	0.534	0.53	0.534	0.53
480	Scrub	0.263	0.26	0.263	0.26
510	Apply Photopolymer	0.672	0.67	0.672	0.67
520	Expose Photopolymer	0.774	0.77	0.774	0.77

530	Develop Photopolymer	0.567	0.57	0.567	0.57
560	Copper/Tin Lead Plate	1.473	1.47	1.473	1.47
580	Mark Control Number	0.445	0.45	0.445	0.45
605	QA Inspect-Tin/Lead	0.184	0.18	0.184	0.18
620	Strip Photopolymer	0.289	0.29	0.289	0.29
680	Etch Copper	0.584	0.58	0.584	0.58
710	Bake	0.072	0.07	0.072	0.07
720	Reflow	0.387	0.39	0.387	0.39
770	Stencil	0.867	0.87	0.867	0.87
775	QA Inspect Scanning	0.656	0.66	0.656	0.66
780	Rout Boards and Coupons	1.069	1.07	1.069	1.07
795	QA Inspect	0.397	0.40	0.397	0.40
815	Coupon Prep	1.069	1.07	1.069	1.07
845	QA Inspect Final	16.689	16.69	16.689	16.69
875	QA Electrical Test	0.263	0.26	0.263	0.26
1005	MIR Inspect	0.685	0.69	0.685	0.69

4.5.3 MCM Fabrication

The DICE-MO system was demonstrated and delivered to the Microelectronics Computer Corporation (MCC) in Austin, TX. MCC has developed information models of there chip and multichip products in EXPRESS. This enables the use of these models with DICE-MO. The information models were provided to Raytheon and we have developed a mock enterprise model based on these models. In order to properly model the enterprise at MCC process and resource information is required.

4.6 How To Build a New Product/Process Model

The Manufacturing Optimization system was designed so that you could build process models against various product models. Provided below are the step by step procedures that a user would follow in order to build new product/process model. We tried sample MCM, EDIF, MMACE, and sheet metal Product Models. The sample sheet metal product is being used to demonstrate the procedures below.

1. Write a Product EXPRESS model to represent the new product area that you wish to build a Process Model against. An small example of a sheet metal product features is provided below (sheetMetal.exp) :

```

SCHEMA sheetMetal_features_schema;
  ENTITY sheetMetal_feature;
    reference_designator: STRING(50);
    quantity: INTEGER;
  END_ENTITY;

  ENTITY hole

```

ABSTRACT SUPERTYPE OF (ONEOF (rectangle,oval,single_diameter_hole))
SUBTYPE OF (sheetMetal_feature);

 minimum_distance_to_edge_of_bend : STRING(50);
 tightest_lead_in_tolerance_plus_minus : STRING(50);
 tightest_hole_to_hole_tolerance_plus_minus : STRING(50);
END_ENTITY;

ENTITY rectangle
SUBTYPE OF (hole);

 length : STRING(50);
 width : STRING(50);
 radius : STRING(50);
END_ENTITY;

ENTITY single_diameter_hole
SUBTYPE OF (hole);

 minimum_hole_diameter : STRING(50);
 maximum_hole_diameter : STRING(50);
 blind_or_through_hole : STRING(50);
 depth_of_hole : STRING(50);
 drill_holes_at_assembly : STRING(50);
 is_the_bottom_of_the_blind_hole_flat : STRING(50);
 setup_rigidity : STRING(50);
END_ENTITY;

ENTITY oval
SUBTYPE OF (hole);

 nominal_hole_length : STRING(50);
 nominal_hole_width : STRING(50);
 radius : STRING(50);
END_ENTITY;
END_SCHEMA;

2. Run the 'express2c++' compiler on the new EXPRESS product model.
The command to compile the sheet metal example is:
express2c++ -noclasses sheetMetal
3. Run 'rose create' command on the new product schema in order to produce an instantiated version of the product. The command to run rose create on the sheet metal example is: **rose create -o sheetMetal_Test sheetMetal_features_schema**
4. Set the MO product/process model schemas environment variable.
setenv MO_PRODUCT_SCHEMAS sheetMetal_Test

5. Make sure the compiled schema and product design file are located in the ROSE_DB environment variable path, or else update ROSE_DB to include the new directory.
6. Run 'mo', enter the 'Modeler', select the empty template process model, and begin building your Manufacturing Process Model for the new product.

5 Conclusions

The concept behind the Manufacturing Optimization (MO) system is to facilitate a two-tiered team approach to the product/process development cycle where a product design is analyzed by multiple engineers, and the product/process changes are traded concurrently in the product and process domains. The system supports Design For Manufacturing and Assemble (DFMA) with a set of tools to model manufacturing processes, and manage tradeoffs across multiple processes. This approach enables refinement of the product design and manufacturing process through collaboration of product design and process development experts; thereby enabling members of the manufacturing process team to analyze a design with respect to the process or speciality they are representing.

The MO prototype system analyzes a unified product model represented in STEP against an established manufacturing enterprise model. The manufacturing enterprise model captures an abstract representation of the manufacturing processes and resources supported by the enterprise. The enterprise model supports representation of reasoning logic which directly associates features of the product model to the processes and resources of the enterprise. The analysis tool evaluates the reasoning logic and selects the appropriate processes and resources. Captured historical cost, yield and rework data from these processes or similar processes enable the system to estimate the cost, yield and rework rates for manufacturing the product under evaluation.

Three tools developed under the DICE program were evaluated for inclusion in the MO system: 1) The Requirements Manager (RM), a system designed to manage product requirements, specifications and corporate policies to support concurrent engineering. 2) The Project Coordination Board (PCB) which provides support for the coordination of the product development activities in a cooperative environment. 3) The Rensselaer Object System for Engineering (ROSE) is an object data manager that has been developed for engineering applications and enhanced to support the DICE program.

The ROSE DB was evaluated and found to be suitable for storing and managing the data files for the manufacturing processes, operations, and steps, as well as, the various manufacturing analysis results. Therefore, all data used by the MO prototype is stored and managed within ROSE.

The RM was evaluated and found to be capable of modeling and evaluating the manufacturing guidelines. The plan was for the RM and the MO software to be coupled through an Application Programming Interface (API) to provide an integrated means of analyzing the manufacturing guidelines, but the two-way API never became available.

The PCB was evaluated as a means to support the communication of the product/process development activities within the product design cycle. A simple model was entered into the PCB as a means of evaluating its viability. The PCB proved too immature a software product to be of added value to the MO system.

The primary output of the program is a prototype software system that is capable of analyzing a product model and a manufacturing process model which provides a preliminary list of the manufacturing processes and the cost and yield estimations required to produce the product. The ROSE database from STEP Tools Inc. was the only existing concurrent engineering technology that we evaluated which was included in the final software product.

6 Recommendations

6.1 Introduction

The goal of this program was to develop a prototype software system capable of modeling manufacturing processes and supporting product/process trade-offs based on those models. The software development cycle matured the software beyond the "prototype" stage resulting in a fairly robust software system. The intent of the recommendations is to identify enhancements required to bring the DICE MO system to a fully functional production level and define the additional needed research. The Raytheon team has tested the software against a number of product models including a Raytheon PWB Assembly, a Sheet Metal, the EDIF PWB Model, and the MCM Model developed by MCC under the DICE Program. The system has been demonstrated to industry, DoD representatives, and the ARPA DICE MO Program Manager at the following events: ARPA On Site Review (2/93), the DICE Program Review (2/93), MCC (1/93), DICE Program Review (2/94), and ARPA Electronic Packaging Initiative (EP/I) Workshop (2/94). The recommendations are based on the results of these tests and demonstrations.

6.2 DICE MO Software Enhancements

6.2.1 User Interface - Functions and Presentation

Graphical Process Modeler - A graphics based manufacturing process flow representation editor for building process models and displaying process analysis results. The editor would include features for creating, deleting, and modifying the process flow or elements of the process flow.

Generation Breakdown Modeler - A "tree like" representation of the product model structure which displays the generational relationship (hence generation breakdown) among assemblies, subassemblies, and component parts. This structure would be used to organize the set of the MO analyses which would be run against a design concept, essentially providing a product based view of a multi-tiered manufacturing process. Supporting functions would include visualization of cost, yield, and process flows at each level of the product structure including roll-up of lower level assembly values to the higher levels.

On line documentation. - The software should include on line documentation including context sensitive help, tutorials, and sample models.

6.2.2 Manufacturing Process Modeler/Analyzer Enhancements

Process Modeler Modularity - The manufacturing process modeler should be re-architected to separate the rule specification code from the manufacturing process resources. This would create a more generic rule specification module which could be used to implement any type of rule based reasoning on STEP based objects.

Incremental Process Analysis - This functionality would allow an initial, or previously run analysis to be updated incrementally in response to design iterations. This capability should provide the user with the ability to add, delete or modify manufacturing process model elements interactively.

Production Attribute Support - The current version of MO executes its analysis based on the product model and a process model constructed by the user. The rule construct of the modeler does not handle production planning data such as production run quantity, lot size, raw stock selection, and learning curves elegantly. The technique of creating an EXPRESS based model of product attributes is an available work-around. More work is needed in this area to create a more robust, user friendly implementation.

Rule Specification Functions - The rule specification code of the manufacturing process modeler should be enhanced to create a richer rule syntax. The enhancements include the ability to call a function when a rule evaluates true, a "where clause" capability, an exponential function, and support for reasoning about manufacturing activity elements.

6.2.3 Integration and Interfaces

Simulation Engine Interface - The capability to feed the output of an analysis run into a discrete simulation engine would provide a powerful addition to the current MO analysis.

Spreadsheet -A spreadsheet interface enabling MO analysis data to be accessed by spreadsheet based mode's .

Database Application Programming Interface - The current version of MO is based on the ROSE database. A database API to a commercial DBMS would enable an enterprise integration path.

6.3 Research and Development Recommendations

While the above recommendations focus on development efforts with immediate application, there are several areas of research related to MO that require exploratory research as noted below.

Electro-Mechanical Assembly/Disassembly Support - Representation and modeling of complex assembly/disassembly operations is an area requiring extensive additional research. Research elements include examining the requirements, needs, and approaches of assembly modeling and parametric based CAD systems to provide fully descriptive models for manufacturing assessment, techniques for generating and assessing electro-mechanical assembly process flows, and virtual reality environment based analyses.

Affordability Metrics Rating Formalism - Analysis capability that assesses performance, cost, and schedule related risk of the product and process. This would be implemented by assessing characteristics associated with affordability and providing a cross referenced index to the risk elements.

Factory Control System Feedback - Process model maintenance will become a critical production issue as model proliferate. The process models share common elements with process planning systems, and shop floor data collection systems. A mechanism must be developed for harmonizing this factory data so that it can be economically maintained. This research should consider standards based activities, such as STEP, and enterprise integration techniques.

7 References

1. Raytheon Company, BR-20558-1, 14 June 1991, DARPA Initiative In Concurrent Engineering (DICE) Manufacturing Optimization - Volume I - Technical.
2. Raytheon Company, Description of CE Technology for the Manufacturing Optimization (MO) System, March 1992.
3. Raytheon Company, Operational Concept for the Manufacturing Optimization (MO) System, March 1992.
4. Raytheon Company, Design Specification for the Manufacturing Optimization (MO) System, December 1992.
5. Raytheon Company, System Description Document for the Manufacturing Optimization (MO) System, July 1993.
6. Raytheon Company, User Manual for the Manufacturing Optimization (MO) System, September 1993.

8 Notes

8.1 Acronyms

API	Application Programming Interface
ARPA	Advanced Research Projects Agency
CAD	Computer Aided Design
CAEO	Computer Aided Engineering Operations
CDRL	Contract Data Requirements List
CERC	Concurrent Engineering Research Center
DBMS	Database Management System
DFMA	Design for Manufacturing and Assembly
DICE	DARPA Initiative In Concurrent Engineering
EDIF	Electronic Data Interchange Format
ISO	International Standards Organization
MO	Manufacturing Optimization
MCC	Microelectronics Computer Corporation
MCM	Multi-Chip Module
OOD	Object Oriented Design
OSF	Open Software Foundation
PCB	Project Coordination Board
PWA	Printed Wiring Assembly
PWB	Printed Wiring Board
PWF	Printed Wiring Fabrication
RAPIDS	Raytheon Automated Placement and Interconnect Design System
RM	Requirements Manager
ROSE	Rensselaer Object System For Engineering
SDAI	STEP Data Access Interface
STEP	Standard for Exchange of Product Model Data

I Manufacturing Enterprise Model

I.1 Process Information Model

The Process Model provides the selection logic for the MO Manufacturing Analyzer when generating the set of process nodes to be included in the cost and yield analysis. The process model is decomposed into a graph of process nodes. Each process node consists of selection rules, dependencies, and operations. Selection rules define criteria that must be satisfied for the process node to be considered in the specific overall manufacturing process. Each process node is dependent on one or more parent nodes. For each process node, an AND/OR flag is kept that specifies if the node is dependent on any one parent node being satisfied, or all parent nodes being satisfied. For a process node to be selected for inclusion in a specific instantiation of a manufacturing process, the following two criteria must be met :

1. At least one of the nodes selection rules is satisfied.
2. All of the nodes parent processes are satisfied or the node AND/OR flag is set to OR and at least one parent node is satisfied.

At each process node there is a list of operations that are performed. Each operation is annotated with an associated yield rate, rework rate, and its usage of resources. From the list of operations the Manufacturing Analyzer will determine the aggregate cost, yield, and rework for this process node.

The EXPRESS model specified in this section was created for process model representation. Figure I-1 is an EXPRESS-G representation of the same model.

I.1.1 EXPRESS Schema for Process Model

This EXPRESS schema listing defines the process model. The process model schema includes two auxiliary schemas, the selection_rules schema and the resource_schema. The specification of these two schemas will follow.

EXPRESS Specification :

*)
INCLUDE 'rules.exp';

```
INCLUDE 'resource.exp';  
  
SCHEMA process_model;  
  
REFERENCE FROM selection_rules;  
REFERENCE FROM resource_schema;
```

(*

I.1.1.1 ProcessModel Entity

A ProcessModel entity is the specification of a manufacturing process model that contains a dependency graph of Process entities. Additional data about the model is also stored including its name, author, creation date, and last modification date.

EXPRESS Specification :

ENTITY ProcessModel;	
name : STRING;	-- Process Model name
creationDate : Date;	-- Model creation date
modifyDate : Date;	-- Model last modify date
author : STRING;	-- Model author
topProcess : Process;	-- Top process in
END_ENTITY;	-- dependency graph

Attribute definitions:

name: Name of the manufacturing process model.

creationDate: The date that the model was created.

modifyDate: The date that the model was last modified.

author: The author of the model.

topProcess: The root or top most process in the process model dependency graph.

I.1.1.2 Process Entity

A Process entity is the specification of a manufacturing process that contains selection rules, operations, and resources. Rules and dependencies of the manufacturing process are modeled in the process nodes. If the rules and dependencies are satisfied, then the process node is included in the overall process analysis. Processes are organized as a directed dependency graph. The dependency graph takes the form of a tree where each node can

have one or more parents and one or more children. Each Process will also have a reference to its immediate siblings (i.e. its neighboring nodes at the same level in the tree).

EXPRESS Specification :

ENTITY Process;	
name: STRING;	-- Process Name
andParents : BOOLEAN	-- AND/OR dependency flag
rules: LIST [0:?] OF ComplexRule;	-- List of Selection Rules
elimrules: LIST [0:?] OF ComplexRule;	-- List of Selection Rules
resources: LIST [0:?] OF Resource;	-- List Of Resources
parents : LIST [0:?] OF Process;	-- List of Parents (Ancestors)
children : LIST [0:?] OF Process;	-- List of Children (Descendants)
lsibling : OPTIONAL Process;	-- Left Sibling
rsibling : OPTIONAL Process;	-- Right Sibling
OperList : LIST [0:?] OF Operation;	-- List of Process Operations
END_ENTITY;	

Attribute definitions:

name: Name of the manufacturing process.

andParents: AND/OR dependency flag. If set to TRUE, the parent nodes to which this node must be satisfied for this node to be considered for selection. If set to FALSE, a minimum of one of the parent nodes must be satisfied for this node to be considered for selection.

rules: List of selection rules. The rules are comprised of design feature entity and attributes being present or of specific values.

elimrules: List of exception rules. These rules are identical to the selection rules in their syntax. If an exception rule is satisfied then this process will not be considered for selection.

resources: List of resources used by the process node as an entity. This list of resources are associated with the process node. A separate list of resources is kept with each operation. Therefore, this list should not contain any resources that have been attached to an operation.

parents: List of processes that this node is dependent on. The nodes in this list are the immediate ancestor to this node. For this node to be selected, depending on the andParents flag, all or one of the nodes in this list must be satisfied.

children: List of processes that are dependent on this node. The nodes in this are direct descendants of this node.

rsibling: The process to the immediate right of this node on the same level of the dependency graph.

Isibling: The process to the immediate left of this node on the same level of the dependency graph.

OperList: List of operations attached to this process. If this node is selected, then the operations in this list will be evaluated for cost, yield and rework.

I.1.1.3 Operation Entity

Attached to each process is a list of operations. When a process is selected and included in the overall manufacturing process, its list of operations is evaluated for cost, rework and yield analysis. Each operation is comprised of the following: A list of resources that are required to perform the operation, the time required to setup for and actually run the operation, an efficiency rate is kept with each operation this provides a factor that when applied to the labor standard for the operation, will calculate the actual time for the operation. Data for computing scrap and rework rates can be stored in a set of lookup tables or specified by a set of rules.

EXPRESS Specification :

*)

```
ENTITY Operation;
  name : STRING;                -- Operation Name
  desc: STRING;                 -- Description
  resources: LIST [0:?] OF ResourceUtilization; -- List Of Resources
  scrap_rate : LIST [0:?] OF Scrap; -- Scrap rates
  rework_rate : LIST [0:?] OF Rework; -- Rework rates
  cost : OPTIONAL OpCost;       -- Operation Cost (for Analyzer)
END_ENTITY;
```

(*

Attribute definitions:

name: Alpha-numeric name of the operation.

desc: Textual description of the operation.

resources: List of resources required to perform the operation.

scrap_rate: A list of a list table entries providing an indexed lookup of scrap rates based on values of entities and their attributes or an equation that when evaluated will provide the scrap rate for the operation.

rework_rate: A list of a list table entries providing an indexed lookup of rework rates based on values of entities and their attributes or an equation that when evaluated will provide the rework rate for the operation.

I.1.1.4 Scrap Entity

The scrap entity is used to represent scrap rate data. Scrap being the percentage of product parts that must be scrapped due to this operation. Scrap data is maintained in a list of scrap entities. In each entity there is a scrap rule and a corresponding scrap rate. If the scrap rule is satisfied, then the corresponding scrap rate is computed.

EXPRESS Specification :

```
*)
    ENTITY Scrap;
        scrapRule : ComplexExp;           -- Rule to be evaluated
        scrapRate : Equation;             -- Scrap that applies if rule is
satisfied
    END_ENTITY;

(*)
```

Attribute definitions:

scrapRule: The scrap rule to be evaluated.

scrapRate: The scrap rate equation to apply if the scrapRule is satisfied.

I.1.1.5 Rework Entity

The rework entity is used to represent rework rate data. Rework being the percentage of product parts that must be reworked due to this operation. Rework data is maintained in a list of rework entities. In each entity there is a rework rule and a corresponding rework rate. If the rework rule is satisfied, then the corresponding rework rate is computed. There is a list of resources associated with the rework which is used to calculate the cost of performing the rework operation.

EXPRESS Specification :

```
*)
    ENTITY Rework;
        reworkRule : ComplexExp;           -- Rule to be evaluated
        reworkRate : Equation;             -- Rework that applies if rule is
satisfied
```

resources : LIST [0:?) OF ResourceUtilization; -- Rework resources
END_ENTITY;

(*

Attribute definitions:

reworkRule: The rework rule to be evaluated.

reworkRate: The rework rate equation to apply if the reworkRule is satisfied.

resources: The resources associated with the rework.

I.1.1.6 OpCost Data

The OpCost data types and entities are used to represent calculated analyzer data associated with an operation.

EXPRESS Specification :

*)

ENTITY OpCost;	
setupTime: REAL;	-- Operation Setup Time
runTime: REAL;	-- Operation Run Time
scrapPercentage: REAL;	-- Actual Calculated Operation Scrap
reworkPercentage: REAL;	-- Calculated Operation Rework
reworkCost: REAL;	-- Calculated Rework Cost
prodQty: INTEGER;	-- Actual Prod. QTY Required
IdealFait: REAL;	-- Calculate Ideal FAIT
ActualFait: REAL;	-- Calculate Actual Estimated FAIT
END_ENTITY;	

(*

Attribute definitions:

setupTime: Operation calculated setup time.

runTime: Operation calculated run time.

scrapPercentage: Operation calculated scrap percentage.

reworkPercentage: Operation calculated rework percentage.

reworkCost: Operation calculated rework cost.

prodQty: Required operation production quantity.

IdealFait: Ideal Operation Fabrication, Assembly, Inspection, and Test Cost

ActualFait: Actual Estimated Operation Fabrication, Assembly, Inspection, and Test Cost

1.1.2 EXPRESS-G Schema for Process Model

The following EXPRESS-G model (figure I-1) represents the Process Model schema:

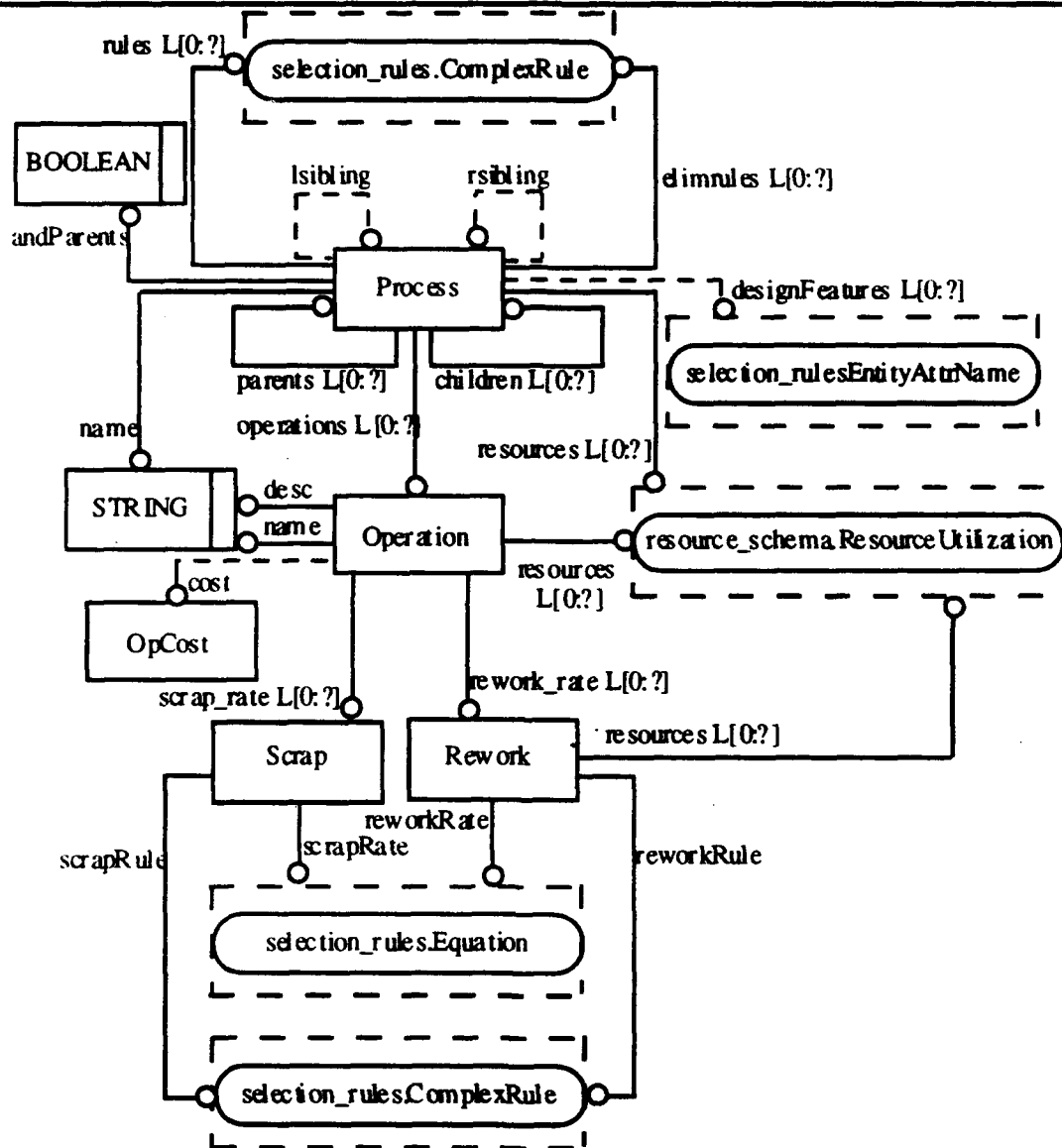


Figure I-1 EXPRESS-G Model of Process Model Schema

I.2 Resource Information Model

I.2.1 EXPRESS Schema for Resource

The resource schema defines a collection of entities that are used to specify resources. A resource is any facility, labor, equipment, or consumable material used in the manufacturing process. A consumable material is a material that is used to aid the manufacturing process and is not considered raw material of the product. As defined in the schema a resource is a generic entity. Specific subtypes of the resource entity are defined to represent facilities, people, equipment, and consumable materials.

EXPRESS Specification :

*)

SCHEMA resource_schema;

I.2.1.1 ResourceUtilization Entity

The ResourceUtilization Entity is used to store which resource(s) are utilized by a process or operation.

EXPRESS Specification :

*)

ENTITY ResourceUtilization;

resource : Resource;

setupTime: Equation;

runTime: Equation;

effRate : OPTIONAL REAL;

END_ENTITY;

-- Resource utilized

-- Setup Equation

-- RunTime Equation

-- Efficiency Rate

(*

Attribute definitions:

resource: The resource being utilized.

setupTime: The amount of setup time required for the resource.

runTime: The amount of time that the resource is being used while running the operation.

effRate: This optional attribute provides an efficiency rate factor that when applied to a labor standard associated with an operation will provide the actual time for the operation.

I.2.1.2 Resource Entity

This is the generic resource entity. Each resource is named and can be coded of a certain type. A list of generic attributes can be attached to each resource using the parameter entity.

EXPRESS Specification :

*)

ENTITY Resource;

resource_name : STRING;

-- Resource Name

resource_code : STRING;

-- Resource Code

parameters : LIST [0:?] of parameter;

-- Resource Parameters

END_ENTITY;

(*

Attribute definitions:

resource_name: The name string associated with the resource.

resource_code: A string used to assign a code to the resource.

parameters: A list of generic attributes that can be attached to this resource.

I.2.1.3 Parameter Entity

The parameter entity is used to define a generic attribute.

EXPRESS Specification :

*)

ENTITY Parameter;

p_name : STRING;

-- Parameter Name

p_value : STRING;

-- Parameter Value

END_ENTITY;

(*

Attribute definitions:

p_name: The name of the parameter.

p_value: The value of the parameter.

I.2.1.4 Labor Entity

The entities in this section define the labor resource. The labor entity is a subtype of the generic resource entity.

EXPRESS Specification :

*)

```
ENTITY Labor SUBTYPE OF (Resource);  
  job_code : STRING;           -- Labor Job Code  
  rate : REAL;                 -- Labor Rate  
END_ENTITY;
```

(*

Attribute definitions:

job_code : A unique identifier associated with the labor.

rate : The labor rate.

I.2.1.5 Equipment Entity

The equipment entity is a subtype of the generic resource entity. It is used to specify the cost of operating the equipment resource during an operation or process.

EXPRESS Specification :

*)

```
ENTITY Equipment SUBTYPE OF (Resource);  
  equipment_category : STRING;  -- Equipment Category  
  cost_per_time_unit : REAL;    -- Cost Per Time Unit  
END_ENTITY;
```

(*

Attribute definitions:

equipment_category: The equipment code or category.

cost_per_time_unit: The cost of operating the equipment resource per unit of time.

I.2.1.6 Facility Entity

The facility entity is a subtype of the generic resource entity. It is used to specify the cost of using the facility resource during an operation or process.

EXPRESS Specification :

*)
 ENTITY facility SUBTYPE OF (Resource);
 square_feet_allocated : REAL; -- Square Feet Allocated
 cost_per_sq_ft_per_time_unit : REAL; -- Cost Per Sq Foot Per Time
 Unit
 END_ENTITY;
 (*

Attribute definitions:

square_feet_allocated: The square feet allocated to this particular operation or process.

cost_per_sq_ft_per_time_unit: The cost per square foot per time unit.

1.2.1.7 ConsumableMaterial Entity

The consumable material entity is a subtype of the generic resource entity. Consumable materials are those materials used to aid in the manufacturing of a product that are consumed by the process. These materials are not considered as part of the raw materials used in the manufacture of the product. They only aid in the production process and are consumed at some measurable rate during the process.

EXPRESS Specification :

*)
 ENTITY ConsumableMaterial SUBTYPE OF (Resource);
 cost_per_unit : REAL; -- Cost Per Unit
 resourceRates: LIST [0:?] OF ResourceConsumable; -- list of resource rates
 END_ENTITY;

 ENTITY ResourceConsumable;
 aresource : Resource; -- Associated Resource
 units_exhausted_per_time_unit : REAL; -- Units Exhausted Per Hour
 END_ENTITY;
 (*

Attribute definitions:

cost_per_unit: The cost of one unit of the consumable material.

resourceRates: The list of resource rates.

aResource: The associated Consumable Resource.

units_exhausted_per_time_unit: Units consumed per unit of time during or by the operation or process.

I.2.2 EXPRESS-G Schema for Resource

The following EXPRESS-G schema (figure I-2) represents the Resource schema:

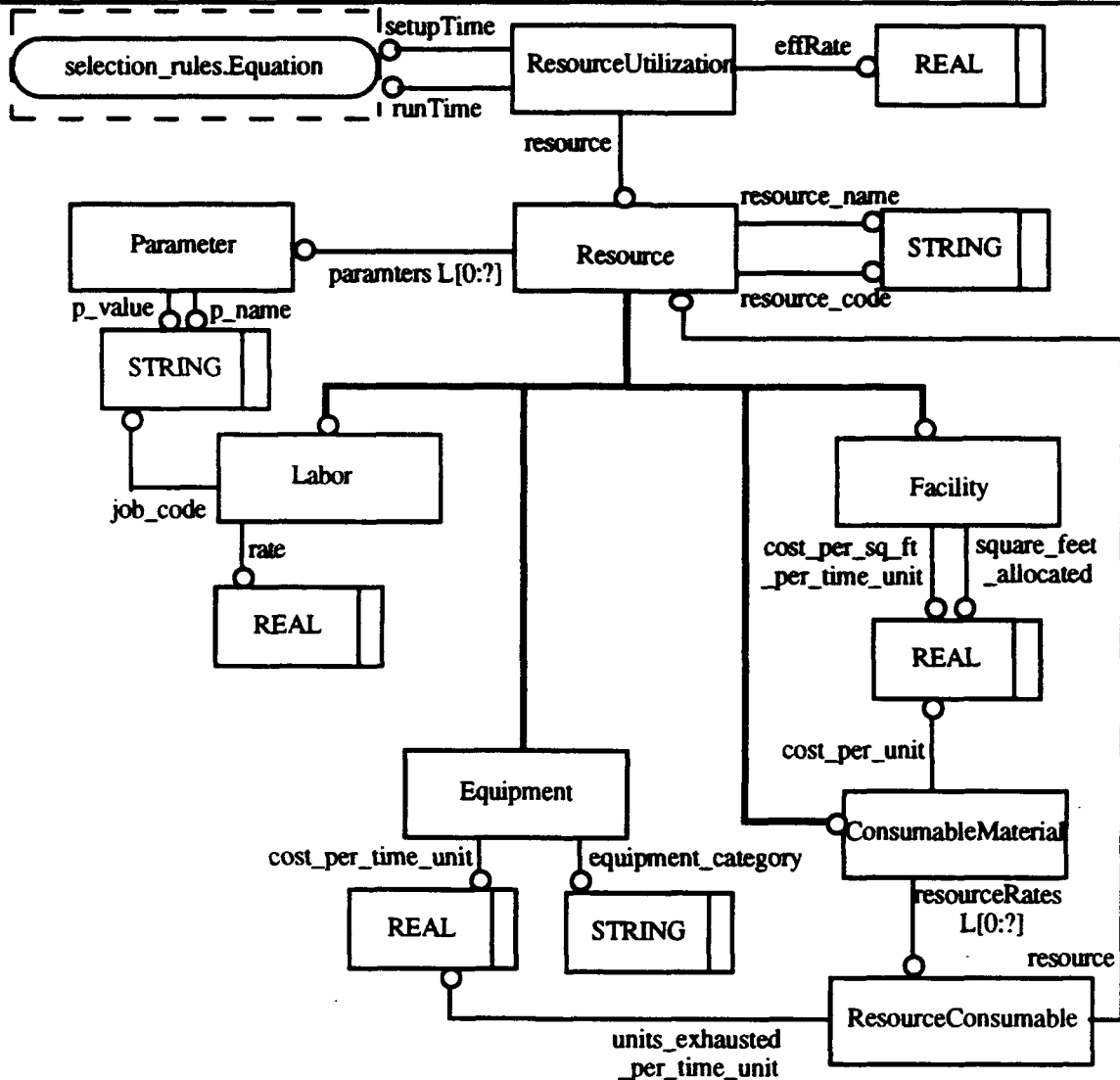


Figure I-2 EXPRESS-G Model of Resources Schema

I.3 Selection Rules Information Model

I.3.1 EXPRESS Schema for Selection Rules

This schema defines a grammar format which rules for selection and equations for evaluation can be specified. Rules are tied to process nodes and equations are tied to such entities as scrap and rework formulas. Provided below is the complete BNF (Backus-Naur

Form) grammar format for the selection rules and equations which the EXPRESS schema is based on.

Complex Rule Grammar Format

<complexRule> := <rules> [<complexRule>]

<rules> := <expression> | <expression> , <rules>

<expression> := <equation> | <equation> <equiv_op> <expression> | <unary_op> <var> | "string"

<equation> := <term> | <term> <op> <equation>

<term> := <const> | <var> | (<equation>)

<var> := <data dictionary strings>

<data dictionary strings> := <alpha> [<alpha> | <digit> | <underscore>]

<const> := real numbers | integers

<op> :=
 * multiplications
 / division
 + addition
 - subtraction

<unary_op> :=
 ! not

<equiv_op> :=
 < less than
 <= less than equal to
 > greater than
 >= greater than equal to
 = equal to
 != not equal to

Operator Precedence (ordered by most --> least priority)

1	!	logical negation	
2	*	multiplications	
	/	division	(left to right)
3	+	addition	
	-	subtraction	(left to right)
4	<	less than	
	<=	less than equal to	
	>	greater than	(left to right)
	>=	greater than equal to	
5	=	equal to	

!= not equal to (left to right)

6 , AND

I.3.1.1 Constants and Types for Rule Construction

The following is a listing of the EXPRESS source that defines symbolic constants and aggregate types that are necessary for the specification of the rules BNF:

EXPRESS Specification :

*)

SCHEMA selection_rules;

```

CONSTANT
  Multiply      : STRING := '*';
  Divide       : STRING := '/';
  Add          : STRING := '+';
  Subtract     : STRING := '-';
  U_Op        : STRING := '!';
  Less        : STRING := '<';
  LessEqual   : STRING := '<=';
  Greater     : STRING := '>';
  GreaterEqual : STRING := '>=';
  Equal       : STRING := '=';
  NotEqual    : STRING := '!=';
  LP          : STRING := '(';
  RP          : STRING := ')';
  Comma       : STRING := ',';
  DQ          : STRING := '"';
END_CONSTANT;

```

```

TYPE DQuote = ENUMERATION OF (DQ);
END_TYPE;

```

```

TYPE AND_Op = ENUMERATION OF (Comma);
END_TYPE;

```

```

TYPE LParen = ENUMERATION OF (LP);
END_TYPE;

```

```

TYPE RParen = ENUMERATION OF (RP);
END_TYPE;

```

```

TYPE Unary_Op = ENUMERATION OF (U_Op);
END_TYPE;

```

```

TYPE Strings = STRING;
END_TYPE;

```

```

TYPE Real_numbers = REAL;
END_TYPE;

```

```

TYPE Integers = INTEGER;

```

END_TYPE;

TYPE

TokenReturnValue = SELECT (Real_numbers, Integers, Strings);
END_TYPE;

TYPE

Const = SELECT (Real_numbers, Integers);
END_TYPE;

TYPE Operator = ENUMERATION OF

(Multiply, Divide, Add, Subtract);
END_TYPE;

TYPE Equiv_Op = ENUMERATION OF

(Less, LessEqual, Greater, GreaterEqual, Equal, NotEqual);
END_TYPE;

(*

1.3.1.2 DataDictStr Entity

The DataDictStr entity is an abstract base class from which two subclass have been created.

The first is the EntityName class which holds the name of an entity name. The other is the EntityAttrName which is used to support the following entity attribute specification :

entity[.attr{.attr[... .attr]]]

An example of an instance of this might be :

line.point1.x

EXPRESS Specification :

*)

ENTITY DataDictStr; -- abstract base class
END_ENTITY;

ENTITY EntityName
SUBTYPE OF (DataDictStr);
name : STRING;
END_ENTITY;

(*

Attribute definitions:

name: The name of the entity as it appears in the product data EXPRESS model.

*)

EXPRESS Specification :

*)

```
ENTITY EntityAttrName
  SUBTYPE OF (DataDictStr);
  entityName : STRING;
  attrName : LIST [1:?] OF STRING;
END_ENTITY;
```

(*

Attribute definitions:

entityName: The name of the entity as it appears in the product data EXPRESS model.

attrName: List of attribute names that correspond to the structure : .attr[.attr[... .attr]].
These attribute name should be specified as they appear in the product data EXPRESS model.

I.3.1.3 ComplexRule Entities

A complex rule is composed of a list of rules. A rule is an Expressions anded together. The following BNF segment defines the grammar of the EXPRESS entities :

<Rules> := <Expression> <AND_OP> | <Expression> <AND_OP> <Rules>

EXPRESS Specification :

*)

```
ENTITY Rules;
  expl : Expression;
  And1 : And_Op;
END_ENTITY;

ENTITY ComplexRule;
  lrule: LIST [0:?] OF Rules;
END_ENTITY;
```

(*

I.3.1.4 Expression Entities

The Expression syntax is represented by the following BNF segment :

<Expression> := <Equation> | <ComplexExp> | <SimpleExp> | <StringValue>

EXPRESS Specification :

*)

TYPE

```
Expression = SELECT (Equation, ComplexExp, SimpleExp, StringValue);  
END_TYPE;
```

```
ENTITY StringValue;  
  quote1 : DQuote;  
  value1 : STRING;  
  quote2 : DQuote;  
END_ENTITY;
```

```
ENTITY ComplexExp;  
  Equ1 : Equation;  
  EquivOp1 : Equiv_Op;  
  Exp1 : Expression;  
END_ENTITY;
```

```
ENTITY SimpleExp;  
  Not1 : Unary_Op;  
  DataDictVar : DataDictStr;  
END_ENTITY;
```

(*

I.3.1.5 Equation Entities

The Equation syntax is represented by the following BNF segment :

<Equation> := <Term> | <ComplexEquation>

EXPRESS Specification :

*)

```
TYPE  
  Equation = SELECT (Term, ComplexEquation);  
END_TYPE;
```

```
ENTITY ComplexEquation;  
  Var1 : Term;  
  Oper1 : Operator;  
  Value : Equation;  
END_ENTITY;
```

```
ENTITY ParenEquation;  
  Lparenthesis : LParen;  
  Equ : Equation;  
  Rparenthesis : RParen;  
END_ENTITY;
```

(*

I.3.1.6 Term Entities

The Term syntax is represented by the following BNF segment :

<Term> := <Const> | <DataDictStr> | <Equation>

EXPRESS Specification :

***)**

TYPE

**Term = SELECT (Const, DataDictStr, Equation);
END_TYPE;**

END_SCHEMA;

(*

I.3.2 EXPRESS-G Schema for Selection Rules

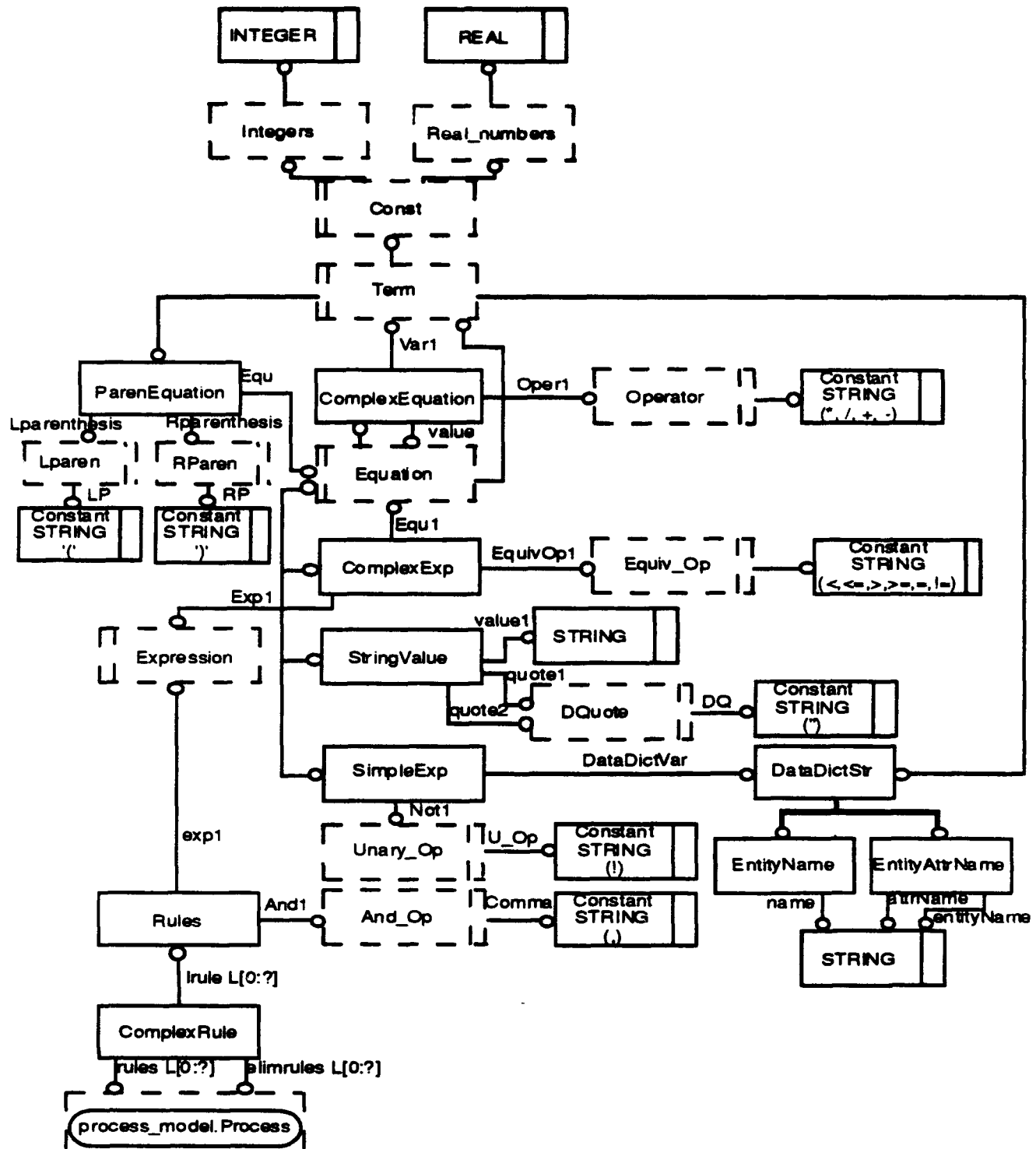


Figure I-3 EXPRESS-G Model of Selection Rules Schema

II PWB Product Information Model

Product data interpretable by the MO system must be modeled in the EXPRESS language and stored as STEP objects in a repository that is interfaced to the STEP Data Access Interface (SDAI). Currently the SDAI only supports a STEP physical file. In the following sections an EXPRESS schema for a PWB product is presented. This schema was created to demonstrate the functionality of the MO system. The schema defines lists of entities that model features of a PWB.

II.1 Printed Wiring Board Product Data Model

At Raytheon, PWB product data is stored in the RAPIDS (Raytheon's Automated Placement and Interconnect Design System) database. Two interfaces were developed to support the transition of PWB product data to and from STEP physical files.

Generating the STEP physical file is facilitated by the interface *RAPIDS to STEP* which maps RAPIDS data items into instantiated STEP entities. We created an information model using the EXPRESS information modeling language. The model was based on the RAPIDS database. The EXPRESS information model was compiled using the STEP Tools *express2c++* compiler which generated a STEP schema and a C++ class library. The class library consists of methods for creating and referencing persistent instances of the STEP entities which are stored in a ROSE database. The STEP schema is used by the STEP Tools *STEP filer* for reading and writing the STEP physical file.

The MO system will use the STEP data directly, as well as for information exchange between the various members of the design team. At Raytheon, the top level team would most likely be using RAPIDS. This is not a requirement for using the core of the MO system. The only requirement is that the top level team and the lower level teams are capable of creating, exchanging and using the STEP physical file.

The Manufacturing Team passes back a consolidated design position to the top level. To aid in the generation of a consolidated position, conflict resolution and design merging must be supported. This is done using the STEP Toolkit from STEP Tools Inc. The *diff* tool reads two versions of a design and creates a delta file. The *difference report generator* reads the

difference file and the original design, and presents each STEP entity and its attributes with the original values and its change state clearly marked with an asterisks.

Once the conflicts of the Manufacturing team members have been resolved, design versions are merged using the STEP Tools *sed* tool. The *sed* tool read the delta file created by the *diff* tool and updates the original design version. This updated version of the design will be transferred back to the top-level product team as the Manufacturing Team's consolidated position.

II.1.1 PWB Design Schema

This is the top level schema for the Raytheon PWB EXPRESS model. The model is primarily derived from the Raytheon's Automated Placement and Interconnect Design System (RAPIDS) data dictionary. RAPIDS is a concurrent engineering design station for Printed Wiring Boards. Its database was designed to capture data from many diverse CAE, CAD, CAM, CAT systems as well as analysis systems for thermal, reliability, critical signal analysis, and manufacturability. Emphasis was placed on making the model extremely modular and flexible.

EXPRESS Specification :

*)

```

INCLUDE 'rpdtypes.exp';
INCLUDE 'rpd_header.exp';
INCLUDE 'alias.exp';
INCLUDE 'annotation.exp';
INCLUDE 'cari.exp';
INCLUDE 'class.exp';
INCLUDE 'comment.exp';
INCLUDE 'dr_block.exp';
INCLUDE 'gate.exp';
INCLUDE 'net.exp';
INCLUDE 'metal_area.exp';
INCLUDE 'part.exp';
INCLUDE 'pin.exp';
INCLUDE 'route.exp';
INCLUDE 'via.exp';
INCLUDE 'xref.exp';
INCLUDE 'shape.exp';
INCLUDE 'stackup.exp';
INCLUDE 'model.exp';

SCHEMA rpd_design;

REFERENCE FROM rpdtypes_schema;
REFERENCE FROM rpd_header_schema;
```

```

REFERENCE FROM alias_schema;
REFERENCE FROM annotation_schema;
REFERENCE FROM cari_schema;
REFERENCE FROM class_schema;
REFERENCE FROM comment_schema;
REFERENCE FROM dr_block_schema;
REFERENCE FROM gate_schema;
REFERENCE FROM net_schema;
REFERENCE FROM metal_area_schema;
REFERENCE FROM part_schema;
REFERENCE FROM pin_schema;
REFERENCE FROM route_schema;
REFERENCE FROM via_schema;
REFERENCE FROM xref_schema;
REFERENCE FROM model_schema;
REFERENCE FROM shape_schema;
REFERENCE FROM stackup_schema;

ENTITY rpd_design_rec;
  alias_header : header_rec;
  aliases : LIST [0:?] of alias_rec;           -- list of aliases
  annotation_header : header_rec;
  annotations : LIST [0:?] of annotation_rec; -- list of annotations
  cari_header : header_rec;
  cari_rules : LIST [0:?] of cari_rule_rec;    -- list of cari rules
  class_header : header_rec;
  classes : LIST [0:?] of class_rec;           -- list of classes
  comment_header : header_rec;
  comments : LIST [0:?] of comment_rec;        -- list of design comments
  dr_block_header : header_rec;
  dr_blocks : LIST [0:?] of dr_block_rec;      -- list of design rule
blocks
  gate_header : header_rec;
  gates : LIST [0:?] of gate_rec;              -- list of gates
  net_header : header_rec;
  nets : LIST [0:?] of net_rec;               -- list of nets
  part_header : header_rec;
  parts : LIST [0:?] of part_rec;             -- list of parts
  pins_header : header_rec;
  pins : LIST [0:?] of pin_rec;              -- list of pins
  route_header : header_rec;
  routes : LIST [0:?] of route_rec;           -- list of routes
  vias_header : header_rec;
  vias : LIST [0:?] of via_rec;              -- list of vias
  xref_header : header_rec;
  xrefs : LIST [0:?] of xref_rec;            -- list of xrefs
  shapes_header : header_rec;
  shapes : LIST [0:?] of pad_shape_rec;       -- list of pad shapes
  stackups_header : header_rec;
  stackups : LIST [0:?] of stackup_rec;       -- list of pad stackups
  models : LIST [0:?] of model_rec;          -- list of part mechanical
models
END_ENTITY;

END_SCHEMA;

```

II.1.2 PWB Generic Types and Entities

This schema defines types and entities that are used throughout the entire PWB model. these types and entities are generic and low level and are used as resources by higher level entities.

EXPRESS Specification:

*)

```
SCHEMA rpdtypes_schema;

TYPE token = STRING; END_TYPE;

TYPE name_type = STRING; END_TYPE;

TYPE layer_type = STRING; END_TYPE;

TYPE keyword = STRING; END_TYPE;

TYPE dimension = INTEGER; END_TYPE;

TYPE shape_type = STRING; END_TYPE;

TYPE loading_type = REAL; END_TYPE;

TYPE blocking_type = STRING; END_TYPE;

-- BINARY data type is not currently supported by the EXPRESS compiler
-- Assuming 8 bit characters (256 layers, 1 bit per layer)
TYPE bitmask = ARRAY [0:31] of STRING(1); END_TYPE;

ENTITY time_rec;
  high : INTEGER;
  low  : INTEGER;
END_ENTITY;

ENTITY r_range_rec;
  minimum : REAL;
  maximum : REAL;
END_ENTITY;

ENTITY i_range_rec;
  minimum : INTEGER;
  maximum : INTEGER;
END_ENTITY;

ENTITY r_span_rec;
  minimum : REAL;
  maximum : REAL;
  span    : REAL;
END_ENTITY;

ENTITY i_span_rec;
  minimum : INTEGER;
  maximum : INTEGER;
```

```

    span : INTEGER;
END_ENTITY;

ENTITY pin_name_rec;
    device : name_type;
    gate   : name_type;
    pin    : name_type;
END_ENTITY;

ENTITY vertex_rec;
    x : dimension;
    y : dimension;
    radius : dimension;
END_ENTITY;

ENTITY point_rec;
    x : dimension;
    y : dimension;
END_ENTITY;

ENTITY loading_rec;
    rated : REAL;
    derated : REAL;
    actual : REAL;
END_ENTITY;

ENTITY attribute_rec;
    key : keyword;
    value : STRING;
END_ENTITY;

END_SCHEMA;

```

II.1.3 Header Data Schema

This schema defines entities for the unit and scale of other entity instances and the creation, access, and modification time entities.

EXPRESS Specification :

```

*),

SCHEMA rpd_header_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY version_rec;
    name : NAME_TYPE;
    revision : NAME_TYPE;
END_ENTITY;

ENTITY header_rec;
    file_name : NAME_TYPE;
    version : NAME_TYPE;
    creation : TIME_REC;
    access : TIME_REC;

```

```

modification : TIME_REC;
unit : NAME_TYPE;
scale : REAL;
tool : NAME_TYPE;
tool_ver : INTEGER;
tool_rev : INTEGER;
assembly : version_rec;
drawing : version_rec;
codeid : NAME_TYPE;           -- Wire Wrap code id
comment : STRING;
attribute : LIST OF ATTRIBUTE_REC;
END_ENTITY;

END_SCHEMA;

```

II.1.4 Alias Data Schema

This is the EXPRESS schema for storing data aliases required by limitations of some CAx system (e.g. NET names in one system are restricted to a particular length that has been violated by a system that is upstream in the design process)

EXPRESS Specification:

```

*)

SCHEMA alias_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY alias_list_rec;
  rapids_name : NAME_TYPE;
  alias_name : NAME_TYPE;
  object_name : NAME_TYPE;
END_ENTITY;

ENTITY alias_rec;
  object : NAME_TYPE;           -- type of object
  property : NAME_TYPE;        -- object property
  system : NAME_TYPE;          -- system requiring an
alias
  alias_list : LIST [0:?] of alias_list_rec; -- list of aliases
  comment : NAME_TYPE;
END_ENTITY;

END_SCHEMA;

```

II.1.5 Annotation Data Schema

This is the EXPRESS model for annotation data. Currently, annotation is limited to text.

EXPRESS Specification:

*)

```

SCHEMA annotation_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY annotation_rec;
    text : STRING;                -- label
    text_height : DIMENSION;      -- text size
    text_width : DIMENSION;       -- text size
    line_width : DIMENSION;       -- width of text line
    layer : NAME_TYPE;            -- text layer
    location : POINT_REC;         -- text location
    rotation : INTEGER;           -- text rotation
    justification : NAME_TYPE;     -- text justification
END_ENTITY;

END_SCHEMA;

```

II.1.6 CARI Data Schema

This Express model is in place for Raytheon legacy data for its proprietary Computer Aided Routing of Interconnect (CARI) system. As a generic model this should be eliminated.

EXPRESS Specification:

```

*)

SCHEMA cari_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY cari_rule_rec;
    cari_id : NAME_TYPE;          -- keyword for CARI record
    record : NAME_TYPE;           -- CARI record card image
    comment : NAME_TYPE;          -- pointer to comment string
END_ENTITY;

END_SCHEMA;

```

II.1.7 Class Data Schema

This EXPRESS model defines data entities for classifying signal nets into groups for particular design rules.

EXPRESS Specification:

```

*)

SCHEMA class_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY class_rec;
    group_name : NAME_TYPE;       -- class identifier

```



```

design_rules : NAME_TYPE;           -- design rules block
signal_list : LIST [0:?] of NAME_TYPE; -- signals in the class
attribute : LIST [0:?] of ATTRIBUTE_REC; -- user defined attribute
comments : LIST [0:?] of STRING;     -- text description
END_ENTITY;

END_SCHEMA;

```

II.1.8 Comment Data Schema

This schema defines a single entity for a comment a list of comments is kept with each PWB design.

EXPRESS Specification :

```

*)

SCHEMA comment_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY comment_rec;
    comment : NAME_TYPE;
END_ENTITY;

END_SCHEMA;

```

II.1.9 Design Rule Data Schema

This EXPRESS schema defines entities for design rules. Design rules are stored in named blocks. Each block except for the GLOBAL block has a Parent name which it inherits from.

EXPRESS Specification :

```

*.

SCHEMA dr_block_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY substrate_block_rec;
    name : NAME_TYPE;           -- substrate name
    technology : NAME_TYPE;     -- technology code
    mode : INTEGER;             -- code for mode
    layers : INTEGER;           -- number of layers
    pad_stack_file : NAME_TYPE; -- RLD file containing pad
                                -- stackups
    layer_model : LIST [0:?] of LAYER_TYPE; -- layer model names
    separation : LIST [0:?] of INTEGER;     -- spacing between layers
    prepreg_mat : NAME_TYPE;               -- prepreg material
    substrate_mat : NAME_TYPE;             -- substrate material
    solder_mat : NAME_TYPE;                -- solder_mask material
END_ENTITY;

END_SCHEMA;

```

```

    attribute : LIST [0:?] of ATTRIBUTE_REC;  -- user defined attributes
END_ENTITY;

ENTITY via_spec_rec;
    via_shape : STRING;                      -- default via shape
    via_length : DIMENSION;                  -- default via length
    via_height : DIMENSION;                  -- default via height
END_ENTITY;

ENTITY via_step_rec;
    via_spacing : DIMENSION;                 -- minimum via separation
    via_depth : INTEGER;                     -- maximum via depth
    first_layer : INTEGER;                   -- first stepping layer
    pattern : NAME_TYPE;                     -- stepping pattern
    direction : REAL;                        -- direction for first step
END_ENTITY;

ENTITY min_space_rec;
    line_to_line : INTEGER;                  -- line-to-line spacing
    line_to_pad : INTEGER;                   -- line-to-pad spacing
    pad_to_pad : INTEGER;                    -- pad-to-pad spacing
    line_to_profile : INTEGER;               -- line-to-profile spacing
    pad_to_profile : INTEGER;                -- pad-to-profile spacing
END_ENTITY;

ENTITY design_block_rec;
    boundary : LIST [0:?] of vertex_rec;    -- design rules boundary
    layer_t : LAYER_TYPE;                    -- design rules layer
    layer_polarity : NAME_TYPE;              -- layer polarity codes
    x_grid : LIST [0:?] of REAL;             -- board routing x grid size
    y_grid : LIST [0:?] of REAL;             -- board routing y grid size
    grid_offset : POINT_REC;                 -- routing grid offset
    x_via_grid : LIST [0:?] of REAL;         -- board via x grid size
    y_via_grid : LIST [0:?] of REAL;         -- board via y grid size
    via_grid_offset : POINT_REC;             -- via grid offset
    spacing : min_space_rec;                 -- feature spacing rules
    via_spec : via_spec_rec;                 -- pointer to default via
    via_stepping : via_step_rec;             -- via stepping data
    acid_trap : INTEGER;                     -- acid trap angle
    attribute : LIST [0:?] of ATTRIBUTE_REC; -- user defined attributes
END_ENTITY;

ENTITY miter_rec;
    angle : DIMENSION;                       -- mitering angle
    length : I_RANGE_REC;                     -- length of miter
END_ENTITY;

ENTITY termination_rec;
    term_type : TOKEN;                       -- type of termination
    (INPUT | OUTPUT | DUAL)
    value : REAL;                             -- resistor value in ohms
    unterm : DIMENSION;                       -- max unterminated length
END_ENTITY;

ENTITY necking_rec;
    line_width : DIMENSION;                  -- minimum necked width
    length : I_RANGE_REC;                    -- length of neck

```

```

    spacing : DIMENSION;
2 necks
END_ENTITY;

ENTITY parallelism_rec;
    parallel_type : NAME_TYPE;
    plane : NAME_TYPE;
    separation : DIMENSION;
between traces
    limit : DIMENSION;
threshold
END_ENTITY;

ENTITY shield_rec;
    shield_type : NAME_TYPE;
microstrip, stripline,
shielded
    signal : NAME_TYPE;
    cover_width : DIMENSION;
    strip_width : DIMENSION;
    isolation : DIMENSION;
    post_spacing : DIMENSION;
    post_stackup : NAME_TYPE;
posts
END_ENTITY;

ENTITY signal_block_rec;
    layers : bitmask;
    layer_t : LIST [0:?] of LAYER_TYPE;
    signal_type : NAME_TYPE;
ground, ecl, etc.
    line_width : DIMENSION;
    line_shape : NAME_TYPE;
    max_length : DIMENSION;
length
    min_length : DIMENSION;
length
    stub : DIMENSION;
    net_order : NAME_TYPE;
DAISY, STAR, WIREWRAP
    route_bias : REAL;
    clearance : DIMENSION;
    place_bias : REAL;
    via_type : NAME_TYPE;
    transmission : DIMENSION;
    span : DIMENSION;
    via_count : INTEGER;
    tolerance : DIMENSION;
    miter : miter_rec;
    termination : termination_rec;
    necking : necking_rec;
    parallelism : LIST [0:?] of parallelism_rec;
    delay_rule : r_span_rec;
    shield_data : shield_rec;
    attribute : LIST [0:?] of ATTRIBUTE_REC;
END_ENTITY;

-- unnecked spacing between

-- total or individual
-- coplanar or biplanar
-- separation threshold

-- parallel traces length

-- shielding type:
-- grounded, guarded,
-- signal shield connected
-- cover width for shield
-- stripline width
-- isolation dist
-- via post space distance
-- stackup for vias for

-- eligible routing layers
-- list of layer types
-- signal type: power,
-- default wire line width
-- line aperture_shape
-- max signal conductor
-- min signal conductor
-- max stub length
-- stringing algorithm: MST,
-- routing priority
-- net isolation distance
-- placement priority
-- pad stack for via
-- max transmission length
-- driver span
-- maximum # of vias
-- matched length tolerance
-- corner mitering rules
-- terminatin rules
-- necking rules
-- parallelism rules
-- propagation delay rules
-- shielding rules
-- user defined attributes

```

```

ENTITY layer_block_rec;
  layer_t : LAYER_TYPE;           -- design rules layer
  cu_weight : REAL;               -- copper weight
  thickness : REAL;               -- thickness of metal
  impedance : INTEGER;            -- layer impedance
  purpose : NAME_TYPE;            -- user define purpose
  attribute : LIST [0:?] of ATTRIBUTE_REC; -- user defined attributes
END_ENTITY;

ENTITY device_block_rec;
  x_grid : LIST [0:?] of REAL;    -- placement grid size
  y_grid : LIST [0:?] of REAL;    -- placement grid size
  grid_offset : POINT_REC;        -- placement grid offset
  layer_name : LAYER_TYPE;        -- component placement layer
  via_flag : BOOLEAN;             -- via inhibit flag
  location_set : NAME_TYPE;       -- placement location set
  auto_insert : NAME_TYPE;        -- auto insertion code
  technology : NAME_TYPE;         -- device technology
  device_bias : REAL;             -- device affinity
  thermal_bias : REAL;            -- thermal affinity
  space_rule : LIST [0:?] OF NAME_TYPE; -- placement spacing rule
  decoupling : DIMENSION;         -- decoupling distance
  overlap : LIST [0:?] OF NAME_TYPE; -- placement overlap rule
  wire_bond : I_RANGE_REC;        -- wire bonding device rules
  aspect : R_RANGE_REC;           -- aspect ratio for resist
  heat_sink : NAME_TYPE;          -- heat sink id
  attribute : LIST [0:?] of ATTRIBUTE_REC; -- user defined attributes
END_ENTITY;

ENTITY metal_area_block_rec;
  pin_clearance : DIMENSION;      -- metal to pin clearance
  via_clearance : DIMENSION;      -- metal to via clearance
  wire_clearance : DIMENSION;     -- metal to wire clearance
  conn_number : INTEGER;          -- connections to each pin
  conn_width : DIMENSION;         -- width of pin connections
  cutout_flag : BOOLEAN;          -- flag to generate cutouts
  suppress_flag : BOOLEAN;        -- unused pad suppression
  show_connect : BOOLEAN;         -- show pad connections
  default_drill : DIMENSION;      -- default drill size
  attribute : LIST [0:?] of ATTRIBUTE_REC; -- user defined attributes
END_ENTITY;

ENTITY dr_block_rec;
  block_name : NAME_TYPE;         -- name of design rule block
  parent_name : NAME_TYPE;        -- name of parent design
rule block
  substrate_block : substrate_block_rec; -- substrate rules
  design_block : design_block_rec;      -- design rules
  signal_block : signal_block_rec;      -- signal rules
  layer_block : layer_block_rec;        -- level rules
  device_block : device_block_rec;      -- signal rules
  metal_area_block : metal_area_block_rec; -- metal area rules
END_ENTITY;

END_SCHEMA;

```

II.1.10 Gate Data Schema

This schema defines entities for device gates.

EXPRESS Specification :

*)

```

SCHEMA gate_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY gate_package_rec;
    component : NAME_TYPE;           -- symbolic component
name
    gate_no    : NAME_TYPE;         -- element number
END_ENTITY;

ENTITY sheet_rec;
    num : NAME_TYPE;                -- sheet number
    x_location : REAL;              -- location on sheet
    y_location : REAL;              -- location on sheet
END_ENTITY;

ENTITY gate_net_rec;
    logic_pin : NAME_TYPE;          -- logical pin name
    signal : NAME_TYPE;             -- default net name
END_ENTITY;

ENTITY gate_rec;
    instance : NAME_TYPE;           -- gate name (handle)
    package : gate_package_rec;     -- package reference
    old_package : gate_package_rec; -- original package ref
    gate_swap_code : NAME_TYPE;     -- swap group name
    swap_inhibit : INTEGER;         -- gate/pin swapability
    gate_count : INTEGER;           -- identical gate/device
    sheet : sheet_rec;              -- schematic location
    comment : NAME_TYPE;            -- pointer to comment
string
    signal_map : LIST [0:?] of gate_net_rec; -- list of pins and nets
    old_signal_map : LIST [0:?] of gate_net_rec; -- list of pins and nets
    attribute : LIST [0:?] of attribute_rec; -- user defined attribute
END_ENTITY;

END_SCHEMA;

```

II.1.11 Net Data Schema

This schema defines entities for net signals.

EXPRESS Specification :

*)

```

SCHEMA net_schema;

```

```

REFERENCE FROM rpdtypes_schema;
REFERENCE FROM pin_schema;
REFERENCE FROM via_schema;
REFERENCE FROM route_schema;
REFERENCE FROM metal_area_schema;
REFERENCE FROM dr_block_schema;

ENTITY ww_pin_data_rec;
  method : NAME_TYPE;           -- installation method
  code : NAME_TYPE;             -- wire type code
  sequence : INTEGER;           -- wrap sequence
  group : NAME_TYPE;            -- wire group
  length : DIMENSION;           -- xs wire length
  findno : NAME_TYPE;           --
  inst_path : STRING;           -- installation path
END_ENTITY;

ENTITY ww_data_rec;
  run_number : INTEGER;         -- wire wrap run number
  func : NAME_TYPE;             -- net function
END_ENTITY;

ENTITY ww_pin_pair_rec;
  method : NAME_TYPE;           -- installation method
  code : NAME_TYPE;             -- wire type code
  sequence : INTEGER;           -- wrap sequence
  group : NAME_TYPE;            -- wire group
  length : INTEGER;             -- xs wire length
  findno : NAME_TYPE;           --
  inst_path : NAME_TYPE;        -- installation path
END_ENTITY;

ENTITY pin_pair_rec;
  t_pin_name : pin_name_rec;    -- to pin name
  f_pin_name : pin_name_rec;    -- from pin name
  t_pin : pin_rec;              -- to pin object
  f_pin : pin_rec;              -- from pin object
  pp_index : INTEGER;           -- index to route object
  pp : route_rec;               -- pointer to route object
  ww_pins : ww_pin_pair_rec;    -- wire wrap pin pair data
END_ENTITY;

ENTITY net_rec;
  name : NAME_TYPE;             -- name of net
  design_rules : NAME_TYPE;     -- design rules block
  signal_type : NAME_TYPE;      -- signal type
  pin_pairs : LIST [0:?] OF pin_pair_rec; -- list of pin pairs
  ww_data : ww_data_rec;        -- wire wrap data
  layer : BITMASK;              -- eligible routing
  layers
    layer_t : LIST [0:?] OF NAME_TYPE; -- list of layer types
    line_width : DIMENSION;           -- line width for
  routing
    line_shape : NAME_TYPE;           -- line aperture_shape
    max_length : DIMENSION;           -- minimum total wire
  length
    min_length : DIMENSION;           -- maximum total wire
  length

```

```

    stub : DIMENSION;                -- maximum stub length
    net_order : NAME_TYPE;           -- stringing algorithm
    clearance : DIMENSION;           -- net isolation
distance
    route_bias : REAL;               -- routing priority
    place_bias : REAL;               -- placement priority
    via_type : NAME_TYPE;            -- absolute pin(via)
type
    transmission : DIMENSION;        -- transmission length
    span : DIMENSION;                -- driver span
    via_count : INTEGER;             -- maximum # of vias
    miter : miter_rec;               -- corner mitering rules
    termination : termination_rec;    -- terminatin rules
    necking : necking_rec;           -- necking rules
    parallelism : LIST [0:?] of parallelism_rec; -- parallelism rules
    shield : shield_rec;             -- shielding rules
    pin_names : LIST [0:?] of pin_name_rec; -- pin names in the net
    pins : LIST [0:?] OF pin_rec;    -- pin records in the
net
    routes : LIST [0:?] of route_rec; -- list of net routes
    vias : LIST [0:?] of via_rec;     -- list of net vias
    metal_areas : LIST [0:?] of metal_area_rec; -- list of net metal
areas
    delay_rule : r_span_rec;          -- propagation delay
rules
    comment : NAME_TYPE;             -- comment string
    attribute : LIST [0:?] OF ATTRIBUTE_REC; -- user defined
attribute
END_ENTITY;

END_SCHEMA;

```

II.1.12 Metal Area Data Schema

This schema defines entities for metal areas (areas of a PWB flooded or meshed with conductor material).

EXPRESS Specification :

SCHEMA metal_area_schema;

REFERENCE FROM rpdtypes_schema;
REFERENCE FROM dr_block_schema;

```

ENTITY cutout_rec;
    cutout_type : NAME_TYPE;          -- type of cutout
    points : LIST [0:?] of POINT_REC; -- cutout description
END_ENTITY;

```

```

ENTITY metal_area_rec;
    signal : NAME_TYPE;
    metal_area_type : NAME_TYPE;      -- type of metal area
    style : NAME_TYPE;                -- style of metal area
    design_rules : dr_block_rec;      -- name of design rule block

```

```

aperture : DIMENSION;           -- apperture for photoplot
spacing : DIMENSION;            -- line spacing in photoplot
layer : INTEGER;                -- layer for metal area
cutout_shape : NAME_TYPE;       -- shape for pin cutouts
origin : POINT_REC;             -- boundary origin
boundary : LIST [0:?] of POINT_REC; -- boundary description
user_cutouts : LIST [0:?] of cutout_rec; -- defined cutouts
auto_cutouts : LIST [0:?] of cutout_rec; -- generated cutouts
comment : NAME_TYPE;            -- comment string
attribute : LIST [0:?] of ATTRIBUTE_REC; -- user defined attribute
END_ENTITY;

END_SCHEMA;

```

II.1.13 Part Data Schema

This schema defines the electrical characteristics of the PWB components.

EXPRESS Specification :

```

SCHEMA part_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY pin_map_rec;
  logic_pin : NAME_TYPE;           -- logical pin name
  component_pin : NAME_TYPE;       -- component pin name
  pin_swap_code : NAME_TYPE;       -- pin swap group
END_ENTITY;

ENTITY element_rec;
  elem_nc : NAME_TYPE;             -- element number
  elem_swap : NAME_TYPE;           -- element Swap Code
  pin_map : LIST [0:?] OF pin_map_rec; -- element to device pin map
END_ENTITY;

ENTITY pec_data_rec;
  rev : NAME_TYPE;                -- pin data rev
  modn : NAME_TYPE;               -- pin data mod
  clear_z : DIMENSION;            -- component CLEARZ
  height : DIMENSION;             -- component HEIGHT
  length : DIMENSION;             -- component LENGTH
  width : DIMENSION;              -- clib component WIDTH
  hsx : DIMENSION;                -- clib HSX pin spacing
  hsy : DIMENSION;                -- clib HSY pin spacing
  mass : REAL;                    -- component MASS
  pin_offset : point_rec;         -- pin offset
END_ENTITY;

ENTITY op_data_rec;
  rev : NAME_TYPE;                -- pin data rev
  modn : NAME_TYPE;               -- pin data mod
  power_dissip : REAL;             -- power dissipation
  max_power_dissip : REAL;        -- max power dissipation
  peak_power : REAL;              -- peak power

```



```

    min_power : REAL;                -- min power
END_ENTITY;

ENTITY therm_data_rec;
    rev : NAME_TYPE;                -- pin data rev
    modn : NAME_TYPE;                -- pin data mod
    emit : REAL;
    rsbtm : REAL;
    rsjb : REAL;
    rsjc : REAL;
    rstop : REAL;
    s_pht : REAL;
    jtm : REAL;
    thermal_type_code : INTEGER;
    thermal_type : NAME_TYPE;
END_ENTITY;

ENTITY pin_time_rec;
    min : REAL;
    typical : REAL;
    max : REAL;
END_ENTITY;

ENTITY input_current_rec;
    iil : REAL;                    -- low current
    iih : REAL;                    -- high current
END_ENTITY;

ENTITY input_voltage_rec;
    vil : REAL;                    -- low voltage
    vih : REAL;                    -- high voltage
END_ENTITY;

ENTITY output_current_rec;
    iol : REAL;
    ioh : REAL;
    iozl : REAL;
    iozh : REAL;
END_ENTITY;

ENTITY output_voltage_rec;
    vol : REAL;                    -- low voltage
    voh : REAL;                    -- high voltage
    vol_min : REAL;                -- min voltage
    voh_max : REAL;                -- max voltage
END_ENTITY;

ENTITY bi_pin_rec;
    input_current : input_current_rec;
    input_voltage : input_voltage_rec;
    output_current : output_current_rec;
    output_voltage : output_voltage_rec;
END_ENTITY;

ENTITY in_pin_rec;
    input_current : input_current_rec;
    input_voltage : input_voltage_rec;
END_ENTITY;

```

```

ENTITY ou_pin_rec;
  ou_config_code : INTEGER;
  ou_config : NAME_TYPE;
  output_current : output_current_rec;
  output_voltage : output_voltage_rec;
END_ENTITY;

ENTITY pin_data_rec;
  rev : NAME_TYPE;
  modn : NAME_TYPE;
  pin_number : NAME_TYPE;
  pin_name : NAME_TYPE;
  pin_swap_code : NAME_TYPE;
  pin_offset : POINT_REC;
  the origin of the device
  capacitance : REAL;
  fall_time : pin_time_rec;
  rise_time : pin_time_rec;
  pin_type : NAME_TYPE;
  bi_pin : bi_pin_rec;
  in_pin : in_pin_rec;
  ou_pin : ou_pin_rec;
END_ENTITY;

ENTITY prop_delay_rec;
  rev : NAME_TYPE;
  modn : NAME_TYPE;
  pin_name_start : NAME_TYPE;
  pin_name_end : NAME_TYPE;
  pin_num_start : NAME_TYPE;
  pin_num_end : NAME_TYPE;
  phl : REAL;
  plh : REAL;
  unateness : NAME_TYPE;
END_ENTITY;

ENTITY part_rec;
  part : NAME_TYPE;
  technology : NAME_TYPE;
  spice_model : NAME_TYPE;
  neat_flag : BOOLEAN;
  stat_flag : BOOLEAN;
  polar_flag : BOOLEAN;
  part_type : NAME_TYPE;
  part_class : NAME_TYPE;
  description : STRING;
  mil_spec : NAME_TYPE;
  findno : NAME_TYPE;
  tolerance : NAME_TYPE;
  value : NAME_TYPE;
  mech_name : NAME_TYPE;
  manufacturer : NAME_TYPE;
  elements : LIST [0:?] OF element_rec;
  geo_data : geo_data_rec;
  op_data : op_data_rec;
  therm_data : therm_data_rec;

```

-- pin data rev
-- pin data mod
-- component pin number
-- component pin name
-- pin swap group name
-- center of the pin relative to

-- rise time
-- fall time
-- B, I, O
-- bi_directional pin data
-- input pin data
-- output pin data

-- pin data rev
-- pin data mod

-- part name
-- device technology
-- spice model for the
device
-- heat sensitivity flag
-- static sensitivity flag
-- polar component flag
-- component type
-- component class
-- component description
-- component mil_spec name
-- component find number
-- component tolerance
-- component value
-- mechanical name
-- part manufacturer
-- list of elements in part
-- geometry data
-- thermal data

```
pin_data : LIST [0:?] OF pin_data_rec;      -- pin data
delay_data : LIST [0:?] OF prop_delay_rec;  -- delay data
comment : NAME_TYPE;                        -- comment string
attribute : LIST [0:?] OF ATTRIBUTE_REC;    -- user defined attributes
END_ENTITY;

END_SCHEMA;
```

II.1.14 Pin Data Schema

This schema defines entities for component pins instantiated on the PWB.

EXPRESS Specification :

*)

```
SCHEMA pin_schema;

REFERENCE FROM rpdtypes_schema;

TYPE function_type = STRING(1) FIXED; END_TYPE;
    -- I for input or source
    -- O output or sink
    -- B bidirectional
    -- T pin on a terminating resistor

ENTITY load_data_rec;
    power : LOADING_TYPE;      -- power loading data
    voltage : LOADING_TYPE;    -- voltage loading data
    current : LOADING_TYPE;    -- current loading data
    temperature : LOADING_TYPE; -- temperature loading data
END_ENTITY;

ENTITY pin_rec;
    pin : NAME_TYPE;          -- pin name
    signal : NAME_TYPE;       -- signal name
    offset : POINT_REC;       -- pin offset from origin
    location : POINT_REC;     -- pin location on board
    rotation : REAL;          -- pin rotation in degrees
    length : BITMASK;         -- pin depth
    suppression : BITMASK;    -- pad suppression mask
    func : FUNCTION_TYPE;     -- pin function code
    stepping : REAL;          -- first stepping direction
    pin_type : NAME_TYPE;     -- absolute pin type
    swap_inhibit : INTEGER;   -- gate/pin swapability
    load_data : load_data_rec; -- pin loading data
    comment : NAME_TYPE;      -- comment string
    attribute : LIST [0:?] of ATTRIBUTE_REC; -- user defined attributes
END_ENTITY;

END_SCHEMA;
```

II.1.15 Conductor Routing Data Schema

This schema defines entities for conductor routes of net signals.

EXPRESS Specification:

*)

SCHEMA route_schema;

REFERENCE FROM rpdtypes_schema;
REFERENCE FROM net_schema;
REFERENCE FROM pin_schema;

```
ENTITY segment_rec;
  x : DIMENSION;           -- x coord of point on the path
  y : DIMENSION;           -- y coord of point on the path
  radius : INTEGER;         -- for circular segment
  segment_width : DIMENSION; -- the width of the segment
END_ENTITY;
```

```
ENTITY ww_route_data_rec;
  revision : NAME_TYPE;     -- wire revision
  sequence : INTEGER;       -- wire wrap sequence
  bends : LIST [0:?] OF POINT_REC; -- wire wrap bend points
END_ENTITY;
```

```
ENTITY route_rec;
  signal : NAME_TYPE;       -- associated signal name
  route_type : NAME_TYPE;   -- type of connection
  status : NAME_TYPE;       -- path status
  target_name : pin_name_rec; -- assigned target pin name
  object_name : pin_name_rec; -- assigned object pin name
  target_pin : pin_rec;     -- assigned target pin
  object_pin : pin_rec;     -- assigned object pin
  target_loc : POINT_REC;   -- coordinates of the target
  object_loc : POINT_REC;   -- coordinates of the object
  protect : BOOLEAN;        -- path protection flag
  target_layer : INTEGER;   -- assigned starting layer
  object_layer : INTEGER;   -- assigned ending layer
  path : LIST [0:?] OF segment_rec; -- list of path segments
  shield_id : INTEGER;      -- code for linking shielding
  pin_pair_index : INTEGER; -- link to pin-pair data
  pin_pair : pin_pair_rec;  -- link to pin-pair data
  ww_data : ww_route_data_rec; -- wire wrapping data
  comment : NAME_TYPE;
END_ENTITY;
```

END_SCHEMA;

II.1.16 Via Data Schema

This schema defines entities for signal net vias.

EXPRESS Specification:

*)

SCHEMA via_schema;

REFERENCE FROM rpdtypes_schema;
REFERENCE FROM dr_block_schema;

REFERENCE FROM net_schema;

```

ENTITY via_rec;
  signal : NAME_TYPE;           -- name of signal net
  location : POINT_REC;         -- board coordinates
  rotation : REAL;              -- via rotation in degrees
  range : BITMASK;              -- pin depth
  suppression : BITMASK;        -- pad suppression mask
  via_type : NAME_TYPE;         -- absolute via type
  via_use : NAME_TYPE;          -- special via use
  shield_id : INTEGER;          -- code for linking
shielding
  shield : shield_rec;          --
  comment : NAME_TYPE;          -- comment string
  attribute : LIST [0:?] of ATTRIBUTE_REC; -- user defined attributes
END_ENTITY;

END_SCHEMA;

```

II.1.17 Library Cross Reference Data Schema

This schema defines entities for the device cross references.

EXPRESS Specification :

*,

SCHEMA xref_schema;

REFERENCE FROM rpdtypes_schema;

REFERENCE FROM pin_schema;

```

ENTITY xref_rec;
  symbolic : NAME_TYPE;         -- symbolic name
  old_symbolic : NAME_TYPE;     -- old symbolic name
  model : NAME_TYPE;            -- mechanical model name
  location : POINT_REC;         -- board location
  mirror : INTEGER;             -- mirror flag
  rotation : REAL;              -- rotation flag
  symbolic_flag : BOOLEAN;      -- symbolic pin names used
flag
  external : BOOLEAN;           -- connector flag
  usa_device : NAME_TYPE;       -- USA device names
  physical : NAME_TYPE;         -- CLIB device name
  raytheon : NAME_TYPE;          -- raytheon part number
  design_rules : NAME_TYPE;     -- design rules block
  layer : NAME_TYPE;            -- component placement layer
  via_flag : BOOLEAN;           -- inhibit via under device
  location_set : NAME_TYPE;     -- placement location set
  auto_insert : NAME_TYPE;      -- auto insertion code
  swap_inhibit : INTEGER;       -- gate/pin swapability code
  fix : BOOLEAN;                -- fixed placement flag
  device_bias : REAL;           -- device affinity
  thermal_bias : REAL;          -- thermal affinity
  coupling : LIST [0:?] of NAME_TYPE; -- placement coupled devices
  decoupling : INTEGER;         -- decoupling distance
  space_rule : LIST [0:?] of NAME_TYPE; -- placement spacing rule

```

```

overlap : LIST [0:?] of NAME_TYPE;      -- placement overlap rule
heat_sink : NAME_TYPE;                  -- heat sink name
load_data : load_data_rec;              -- loading data
comment : NAME_TYPE;                    -- comment string
attribute : LIST [0:?] of attribute_rec; -- user defined attributes
END_ENTITY;

END_SCHEMA;

```

II.2 PWB Design Data EXPRESS-G Model

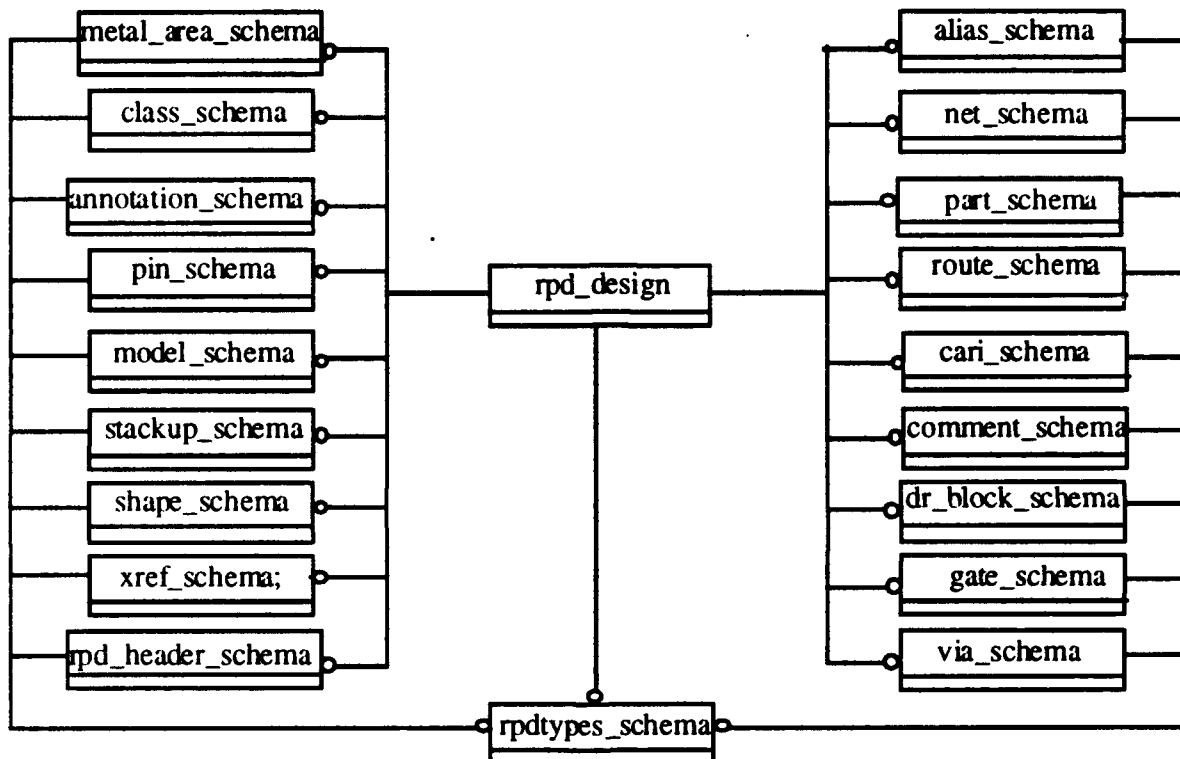


Figure II-1 PWB Schema Level EXPRESS-G Model

II.3 Electronic Component Library Data Model

II.3.1 Component Model Data Schema

This schema defines entities for modeling PWB components.

EXPRESS Specification :

*)

SCHEMA model_schema;

REFERENCE FROM rpdtypes_schema;

REFERENCE FROM rpd_header_schema;

REFERENCE FROM stackup_schema;

```
ENTITY rev_data_rec;
  issue_date : NAME_TYPE;      -- date of issue
  revision : NAME_TYPE;        -- revision number
  eco : NAME_TYPE;             -- latest eco number
  eco_date : NAME_TYPE;        -- date of latest eco
END_ENTITY;
```

```
ENTITY dev_origin_rec;
  origin_type : NAME_TYPE;     -- origin types
  center : POINT_REC;          -- device center
  offset : POINT_REC;          -- placement offset
  mirror : INTEGER;            -- reflection code
END_ENTITY;
```

```
ENTITY label_rec;
  text : STRING;               -- label text
  height : DIMENSION;          -- text size
  width : DIMENSION;           -- text size
  location : POINT_REC;        -- text location
  rotation : INTEGER;          -- text rotation
  line_width : DIMENSION;      -- width of text line
  justify : NAME_TYPE;         -- text justification
END_ENTITY;
```

```
ENTITY boundary_rec;
  boundary_type : NAME_TYPE;    -- type of boundary
  shape : NAME_TYPE;            -- boundary outline shape
  outline : LIST [0:?] of VERTEX_REC; -- boundary outline vertices
  layers : LIST [0:?] of NAME_TYPE; -- boundary layers
END_ENTITY;
```

```
ENTITY obstruction_rec;
  obstruction_type : NAME_TYPE; -- type of obstruction
  shape : SHAPE_TYPE;           -- outline shape
  outline : LIST [0:?] of VERTEX_REC; -- pad outline
  layers : LIST [0:?] of LAYER_TYPE; -- pad layers
  blocking : LIST [0:?] of BLOCKING_TYPE; -- blocking codes
END_ENTITY;
```

```
ENTITY device_rec;
  symbolic : NAME_TYPE;         -- symbolic name
  physical : NAME_TYPE;         -- physical name
  model : NAME_TYPE;            -- mechanical model name
  location : POINT_REC;         -- location on board
  rotation : REAL;              -- rotation in degrees
  mirror : INTEGER;             -- mirror flag
END_ENTITY;
```

```
ENTITY dev_pin_rec;
  physical : STRING;             -- physical pin name (must be
string of integers)
  symbolic : NAME_TYPE;         -- symbolic pin name
  location : POINT_REC;         -- pin location
  drill : DIMENSION;            -- default drill size
  stackup_name : NAME_TYPE;     -- pad stackup name
  stackup : STACKUP_REC;        -- pad stackup record
```

```

rotation : REAL;                -- stackup rotation
offset : POINT_REC;            -- stackup offset
stepping : INTEGER;            -- first stepping direction
END_ENTITY;

ENTITY thermal_rec;
  thermal_type : NAME_TYPE;      -- type of thermal relief
  width : DIMENSION;            -- line width
  spacing : DIMENSION;          -- line spacing
  stackup_name : NAME_TYPE;      -- stackup name
  stackup : STACKUP_REC;        -- stackup record
END_ENTITY;

ENTITY package_rec;
  package_type : NAME_TYPE;      -- package type
  category : NAME_TYPE;         -- package category
  orientation : NAME_TYPE;      -- package orientation
  distance : DIMENSION;         -- pin row separation
  depth : DIMENSION;           -- package depth
  height : DIMENSION;          -- package height
  width : DIMENSION;           -- package width
  lead : DIMENSION;            -- package lead diameter
  fix : BOOLEAN;               -- fixed device flag
  body_diameter : DIMENSION;    -- package body diameter
  span : DIMENSION;            -- package pin span
  insert : NAME_TYPE;          -- package insertion code
  mechanical : BOOLEAN;        -- mechanical device flag
  auto_ww_offset : POINT_REC;    -- automatic wirewrap offset
  auto_ww_trp : INTEGER;        -- automatic wirewrap initial
  trp
    semi_ww_offset : POINT_REC;  -- semiautomatic wirewrap offset
    semi_ww_trp : INTEGER;       -- semiautomatic wirewrap
  initial trp
END_ENTITY;

ENTITY model_rec;
  header : header_rec;          -- pointer to header
  record
    mn_name : NAME_TYPE;        -- mechanical model name
    rev_data : rev_data_rec;    -- revision data
    origin : dev_origin_rec;    -- origin data
    package : package_rec;      -- packafing data
    labels : LIST [0:?] of label_rec; -- list of labels
    boundaries : LIST [0:?] of boundary_rec; -- list of boundaries
    obstructions : LIST [0:?] of obstruction_rec; -- list of obstructions
    devices : LIST [0:?] of device_rec; -- list of devices
    pins : LIST [0:?] of dev_pin_rec; -- list of pins
    thermals : LIST [0:?] of thermal_rec; -- list of thermal
  reliefs
    comments : LIST [0:?] of STRING; -- list of comments
    attribute : LIST [0:?] of attribute_rec; -- list of user defined
  attributes
END_ENTITY;

END_SCHEMA;

```


II.3.2 Pad Stack Data Schema

This schema defines entities for pin and via pad stackups. Various pad shapes for each layer are combined. The layer assignments are then combined to form the padstack.

EXPRESS Specification:

*)

```

SCHEMA stackup_schema;

REFERENCE FROM rpdtypes_schema;
REFERENCE FROM shape_schema;

ENTITY pad_rec;
    pad_name : NAME_TYPE;           -- shape name
    pad_shape : PAD_SHAPE_REC;      -- pad shapes
    func : NAME_TYPE;              -- pad function
END_ENTITY;

ENTITY pad_stack_rec;
    model : NAME_TYPE;             -- layer model
    offset : POINT_REC;            -- pad offset
    pad_list : LIST [0:?] of pad_rec; -- pad_names
END_ENTITY;

ENTITY stackup_rec;
    stack_name : NAME_TYPE;        -- name of stackup
    pad_stack : LIST [0:?] of pad_stack_rec; -- pad stackups
    drill : INTEGER;              -- default drill size
    comments : LIST [0:?] of STRING; -- list of comments
END_ENTITY;

END_SCHEMA;

```

II.3.3 Pad Shape Data Schema

This schema defines entities for pin and via pad shapes.

EXPRESS Specification:

*)

```

SCHEMA shape_schema;

REFERENCE FROM rpdtypes_schema;

ENTITY shape_rec;
    shape : NAME_TYPE;           -- shape type
    width : DIMENSION;          -- aperature width
    outline : LIST [0:?] of VERTEX_REC; -- shape description
END_ENTITY;

ENTITY pad_shape_rec;
    name : NAME_TYPE;           -- shape name

```

```
pads : LIST [0:?] of shape_rec;      -- pad shapes
END_ENTITY;

END_SCHEMA;
```

II.4 Electronic Component Library Data EXPRESS-G Model

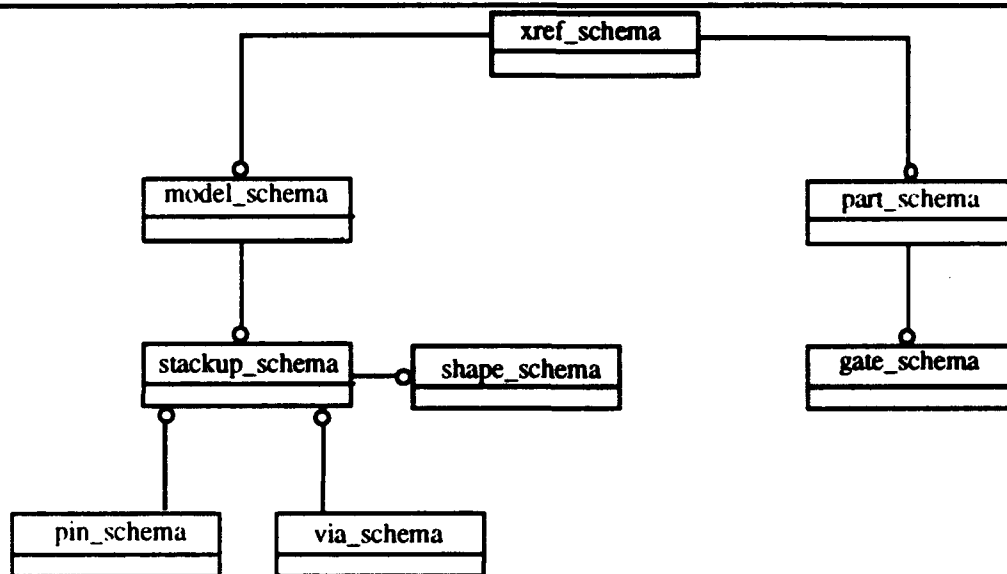


Figure II-2 Component Data EXPRESS-G Schema

III User Interface Screens

The main user interface window for MO provides access to the various modules within the system, including the product and process STEP files, the manufacturing analyzer, the manufacturing advisor, the process modeler, and system help. Figure III-1 depicts the MO main window.

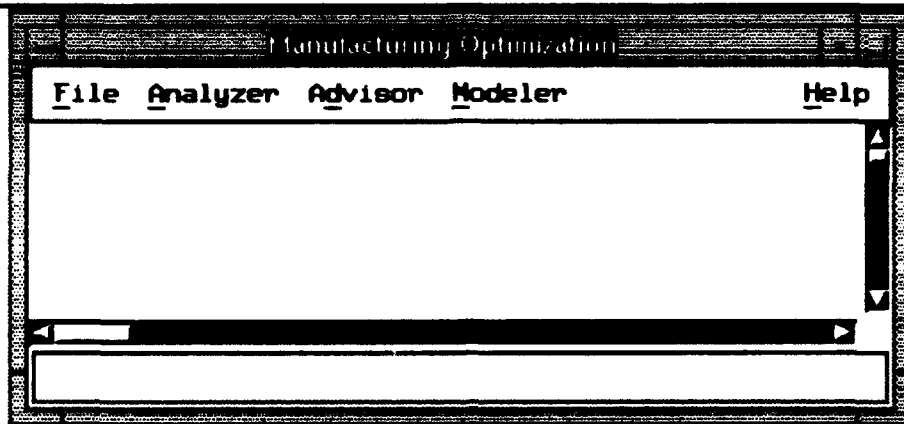


Figure III-1 MO Main Window

III.1 File Menu

The File menu provides a means to select and edit the product and process data and provides access to two translators. Rapids2Step translates PWB design data from a Raytheon propriety format to STEP. Step2Rapids translated a PWB design from STEP to a Raytheon propriety format. Figure III.1 illustrates the MO main window with the File menu pulled down.

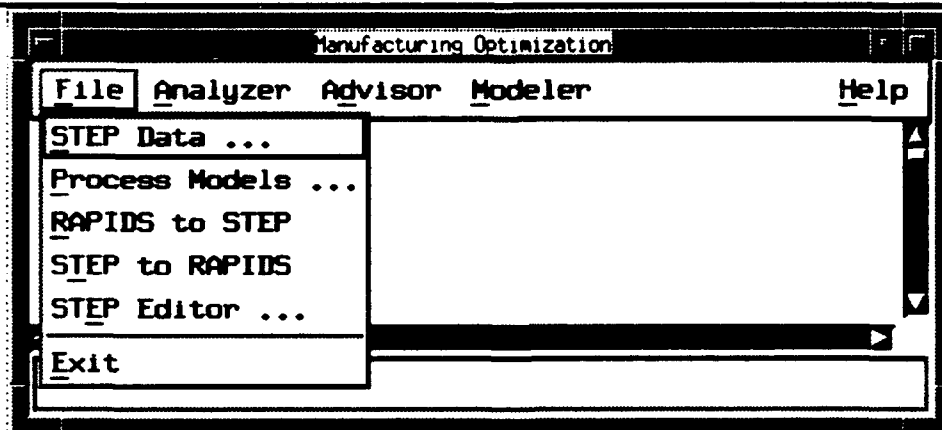


Figure III.1 File Options

III.1.1 Product/STEP Data Selection

MO allows the user to select a product/STEP data file for analysis. When the STEP Data button is selected, Figure III.1.1 is displayed. A user performs a selection for choosing a design database to analyze.

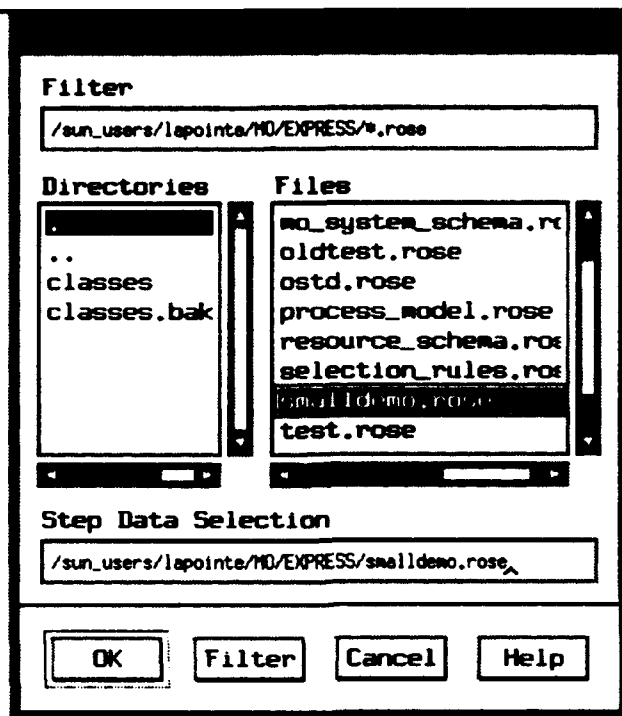


Figure III.1.1 Product Data Selection/Edit Menu

III.1.2 Process Model Selection

MO allows the user to select a process model for use during analysis. When the process model button is selected, Figure III.1.2 is displayed.

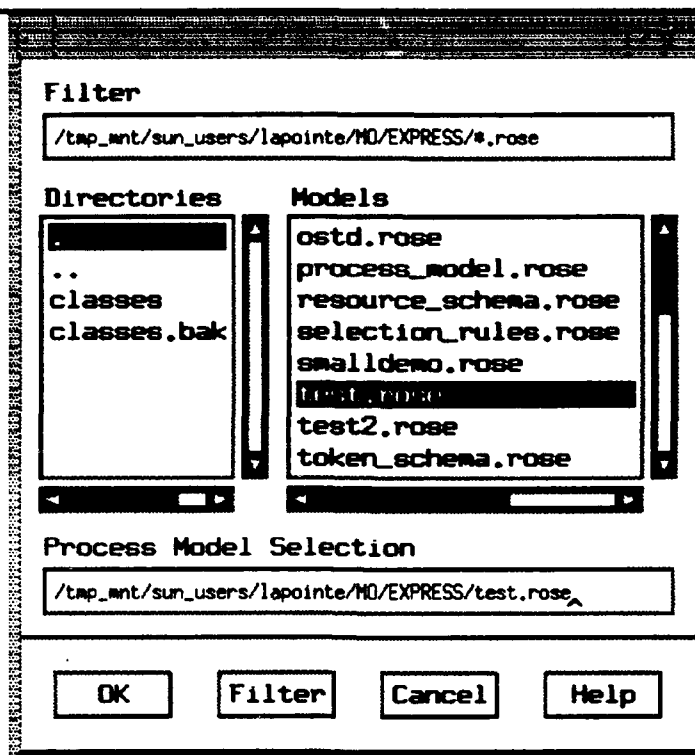


Figure III.1.2 Process Model Selection Menu

III.1.3 RAPIDS to STEP Translator Interface

Rapids2step is a C++ application that utilizes the ROSE database and tools developed by STEP Tools Inc. The program reads the RAPIDS Structured and Library Databases (RSD/RLD) using the RAPIDS Procedural Interface. Once all of the records have been read from the RSD and RLD databases and the corresponding STEP object lists have been created, the STEP file is created and the STEP objects are written to it by the ROSE STEP filer. The MO system provides the user with an interface to the rapids2step translator. The interface is shown in Figure III.1.3.

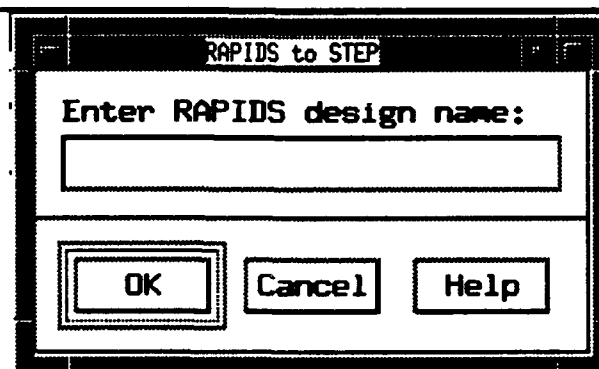


Figure III.1.3 RAPIDS to STEP Form

III.1.4 STEP to RAPIDS Translator Interface

Step2rapids is also a C++ application that utilizes ROSE and tools developed by STEP Tools Inc. The program reads a STEP file conforming to the EXPRESS schemas developed as part of this project. The ROSE STEP filer is used to read the STEP file into instances of classes created by the express2c++ compiler. Each of the STEP object lists is traversed and for each object in the list an appropriate C structure corresponding to the RAPIDS procedural interface is created and its fields are populated with the values of the corresponding attributes of the STEP object. The MO system provides the user with an interface to the step2rapids translator. The interface is shown in Figure III.1.4.

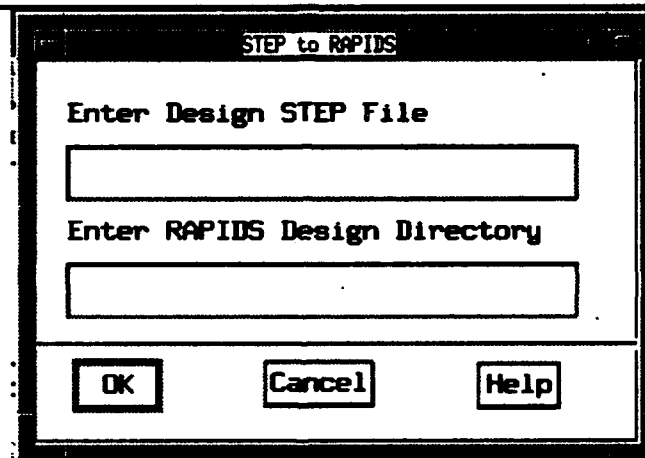


Figure III.1.4 STEP to RAPIDS Form

III.1.5 STEP Editor

The MO system provides direct access to the STEP Tools, Inc. stepeditor tool. The STEP Editor enables the user to add, delete and modify STEP entity instances.

III.2 Analyzer Form

The MO system provides the user with the ability to perform a manufacturability analysis based on a selected manufacturing process model versus a particular product design database through the analyzer button on the main window. The analyzer determines the appropriate processes required to build the product based on the selected process model, calculates the overall yield and rework rates of processes, operations, and steps based on the selected process flow,

calculates the ideal time to perform the processes, operations, and steps. The yield rates are incorporated to project the estimated actual times. The cost utilizes the ideal and estimated actual labor times by multiplying them with the resource(s) labor rate(s) to obtain the ideal and estimated actual cost of each process, operation, and step, as well as the cost of the entire part. When a user selects the analyzer button, the system begins the cycle of selecting the applicable processes, calculating the yield and rework, and finally to determine the ideal and actual estimated cost of the part under analysis. The user can then select the type(s) of analysis to be performed.

III.3 Advisor Window

The Manufacturing Advisor module provides the capability of viewing the analyzer results. The user can select analysis runs to view. The user can display process, quality, or costing results as graphs, and can also view complete analysis data to the screen or to file in report format. Figure III.3 illustrates the Manufacturing Advisor window which is displayed when the user selects the Advisor button on the MO Main Window.

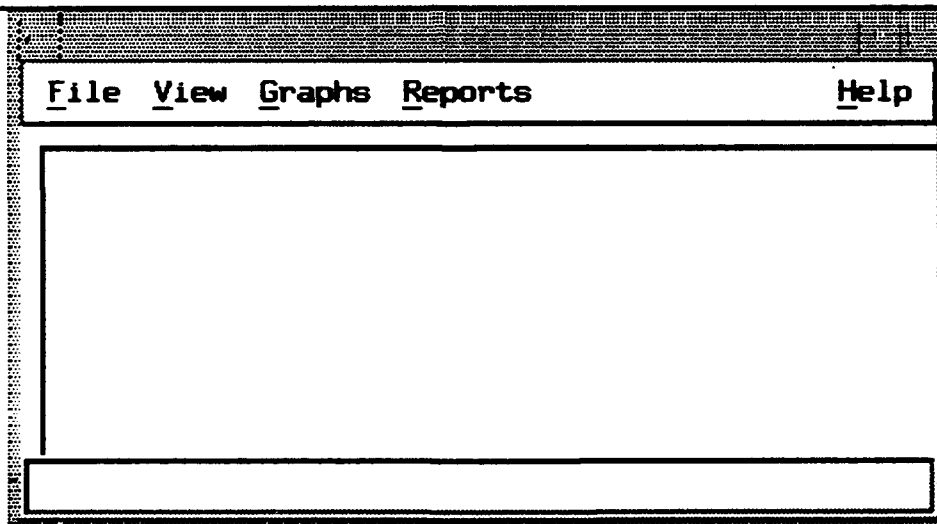


Figure III.3 Manufacturing Advisor Window

III.3.1 File

The MO system supports viewing of one or more analysis runs so that the user can visually see the results, as well as visually compare analyses. The user can select the run(s) which he/she wants to view. The user may select the Exit button to leave the Advisor window. Figure III.3.1 illustrates the Advisor window with the File menu pulled down.

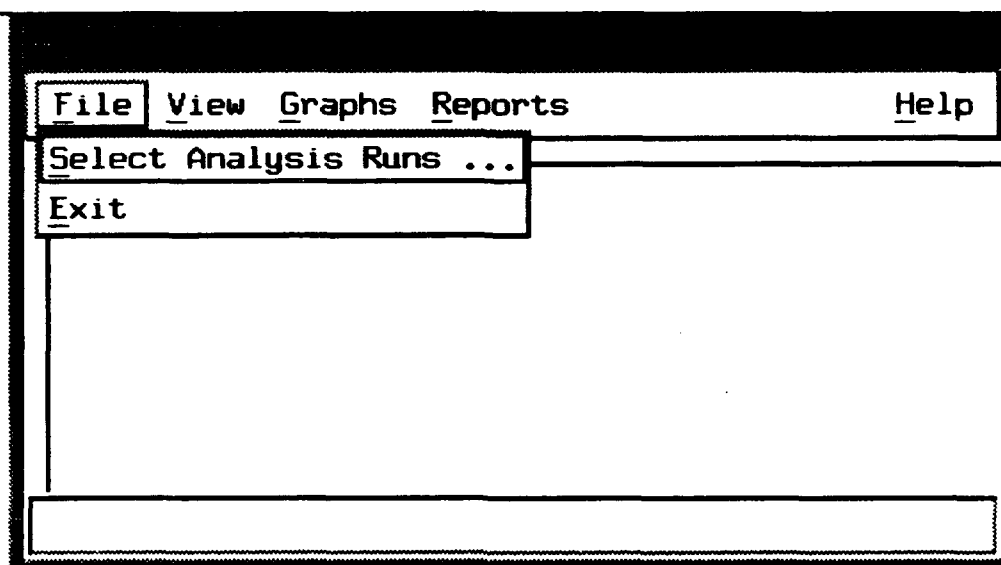


Figure III.3.1 File menu

III.3.2 View

The View pull-down menu contains a series of toggle buttons where the user can specify the level of activity to view. The analyzer results can be viewed in several ways: processes, operations, steps, or all activities. All activities is the default selection. Figure III.3.2 illustrates the Advisor window with the View menu pulled down.

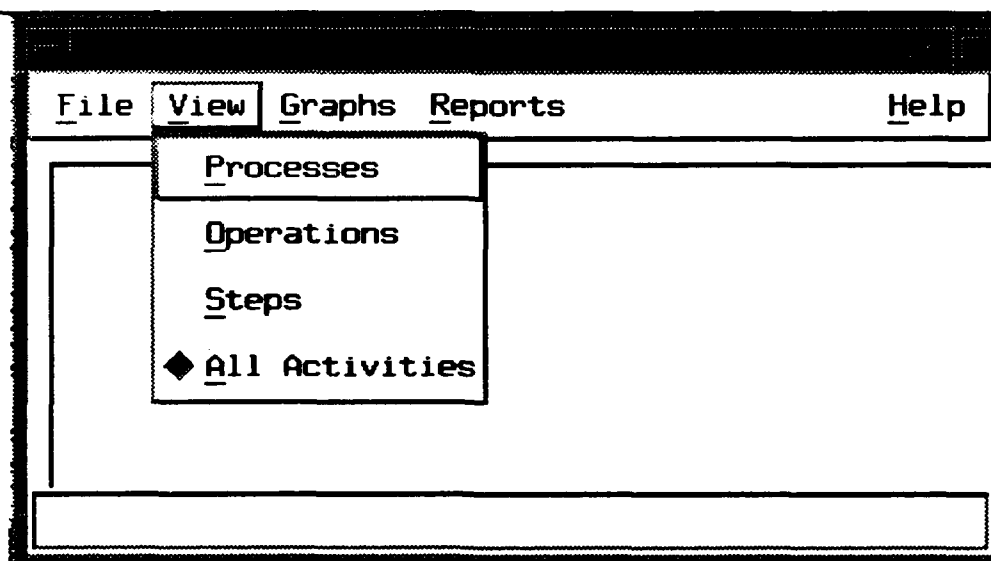


Figure III.3-2 View menu

III.3.3 Graphs

Results of the analysis runs can be viewed in three formats: activity, quality, and cost graphs. The Graphs pull-down menu is shown in Figure III.3.3-1.

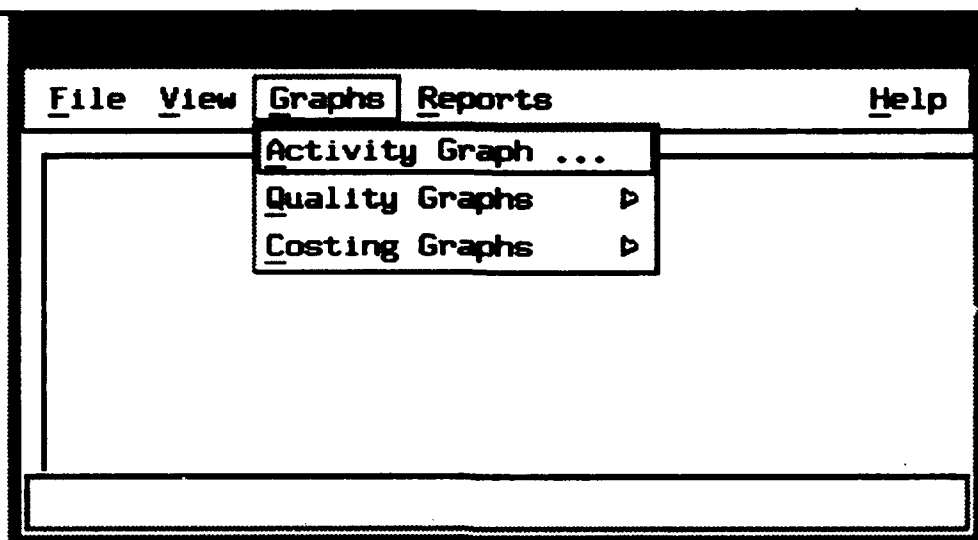


Figure III.3.3-1 Graphs menu

III.3.3.1 Activity Graphs

When the Activity Graph button is chosen, the selected analysis run(s) process flow is graphically displayed. Figure III.3.3.1 illustrates the resulting process flow graph for one set of analysis results.

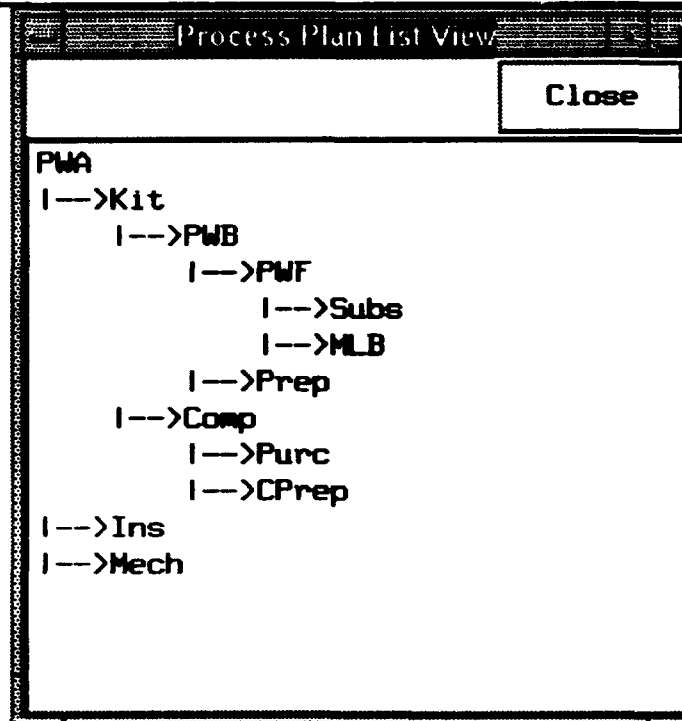


Figure III.3.3.1 Process Plan List View Form

III.3.3.2 Quality Graphs

The MO system provides for graphically displaying the quality results associated with analysis runs including graphs for yield, rework, and production quantity. Figure III.3.3.2-1 illustrates the Advisor Window with the Quality Graphs menu pulled down.

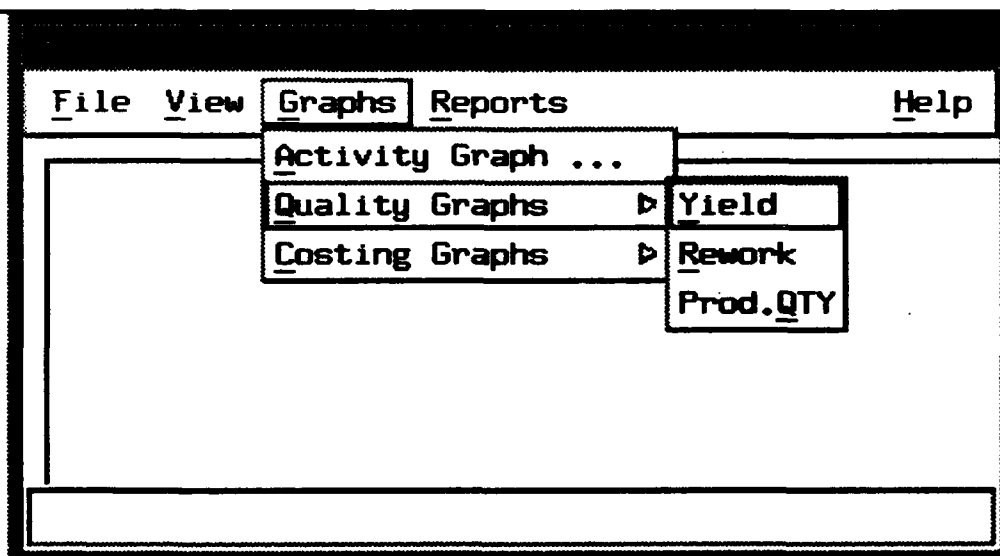
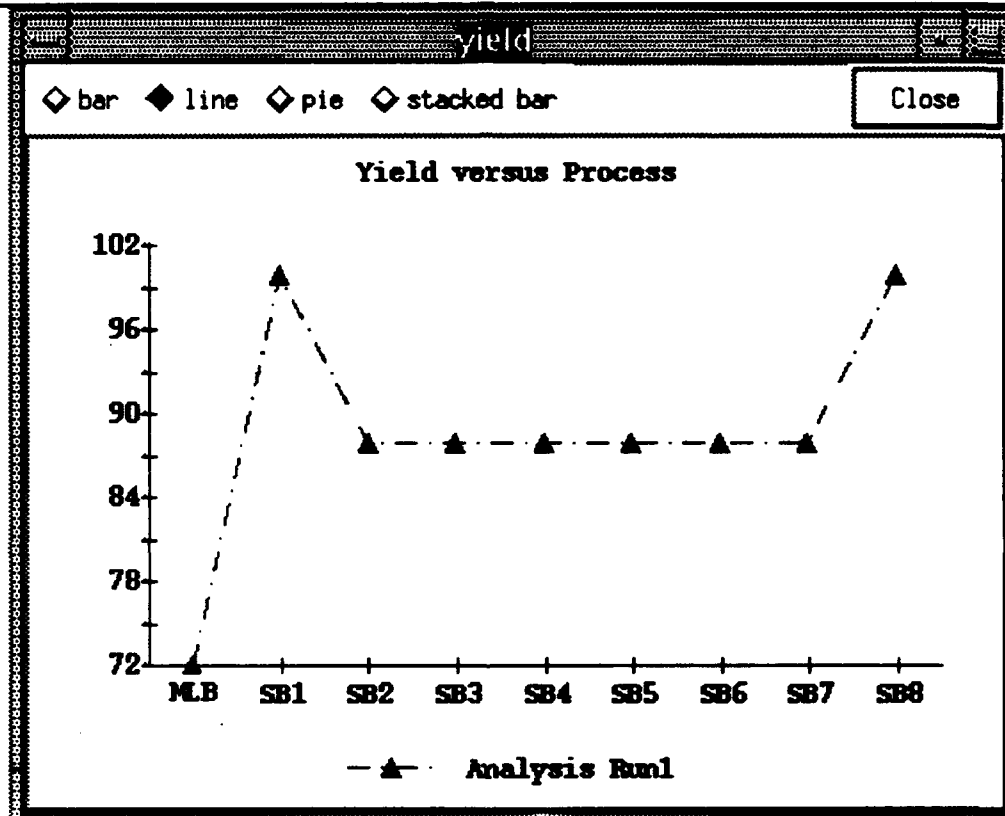


Figure III.3.3.2-1 Quality Graphs

The graphs are also displayed in a separate window where the user can select to display the data as a bar, stacked bar, line, or pie chart. The yield and production quantity defaulting display is a line chart, and the rework is a bar chart. Figure III.3.3.2-2 illustrates a sample yield versus process line graph.

**Figure III.3.3.2-2 Yield versus Process Graph****III.3.3.3 Costing Graphs**

The MO system provides for graphically displaying the costing results associated with analysis runs including graphs for ideal/actual time and rework/ideal/actual cost. Figure III.3.3.3-1 illustrates the Advisor Window with the Costing Graphs menu pulled down.

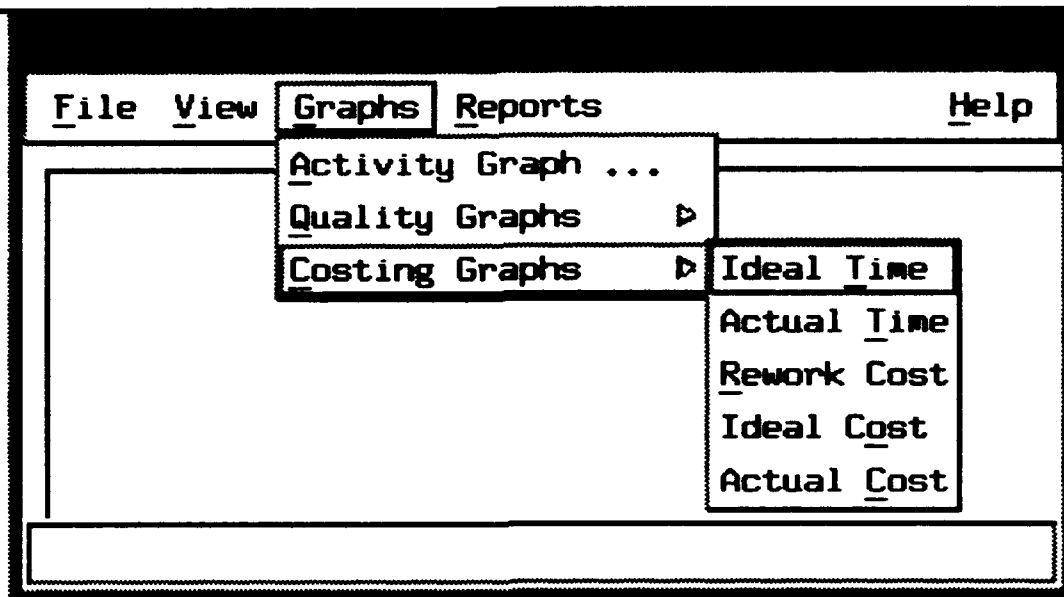


Figure III.3.3.3-1 Costing Graphs

The costing graphs are displayed in a separate window where the user can select to display the data as a bar, stacked bar, line, or pie chart. The time default display will be a line chart, and the cost default display will be a bar chart. Figure III.3.3.3-2 illustrates a sample actual time versus manufacturing activity line graph.

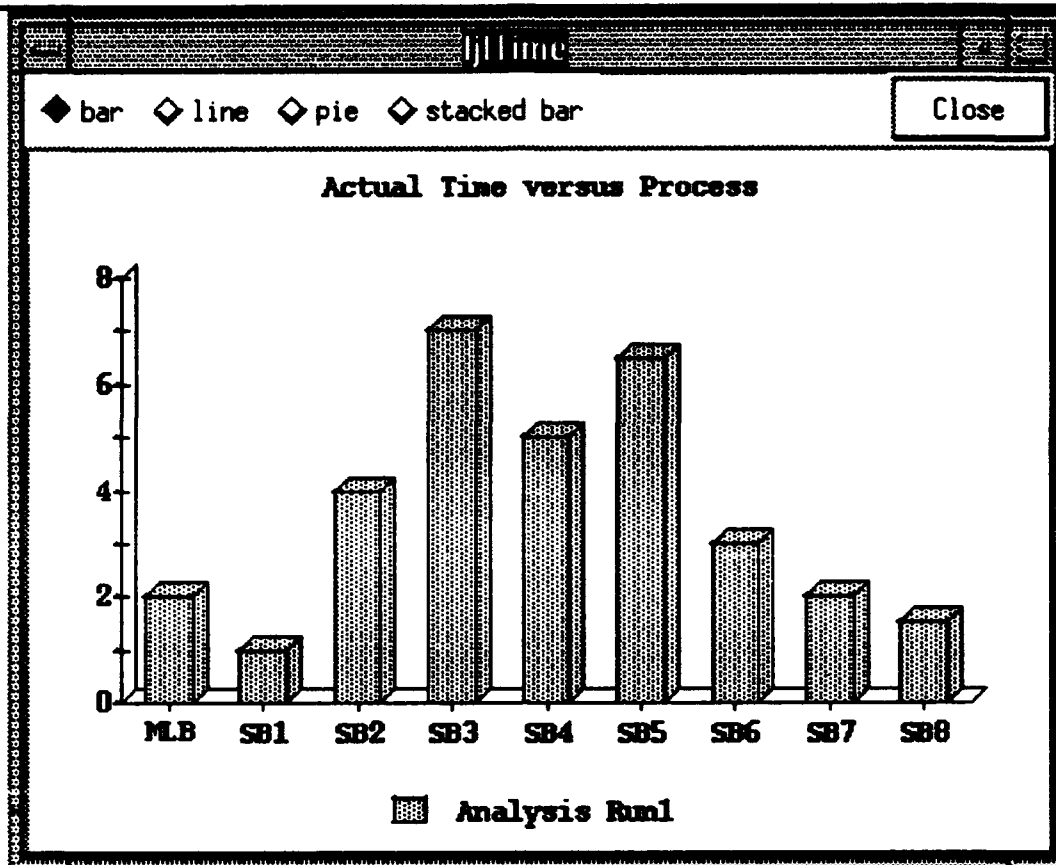


Figure III.3.3.3-2 Actual Time versus Process Graph

The user may click on any point in the line graph and a small pop up window will appear displaying the exact value for the specified point on the graph. Figure III.3.3.3-3 illustrates this point.

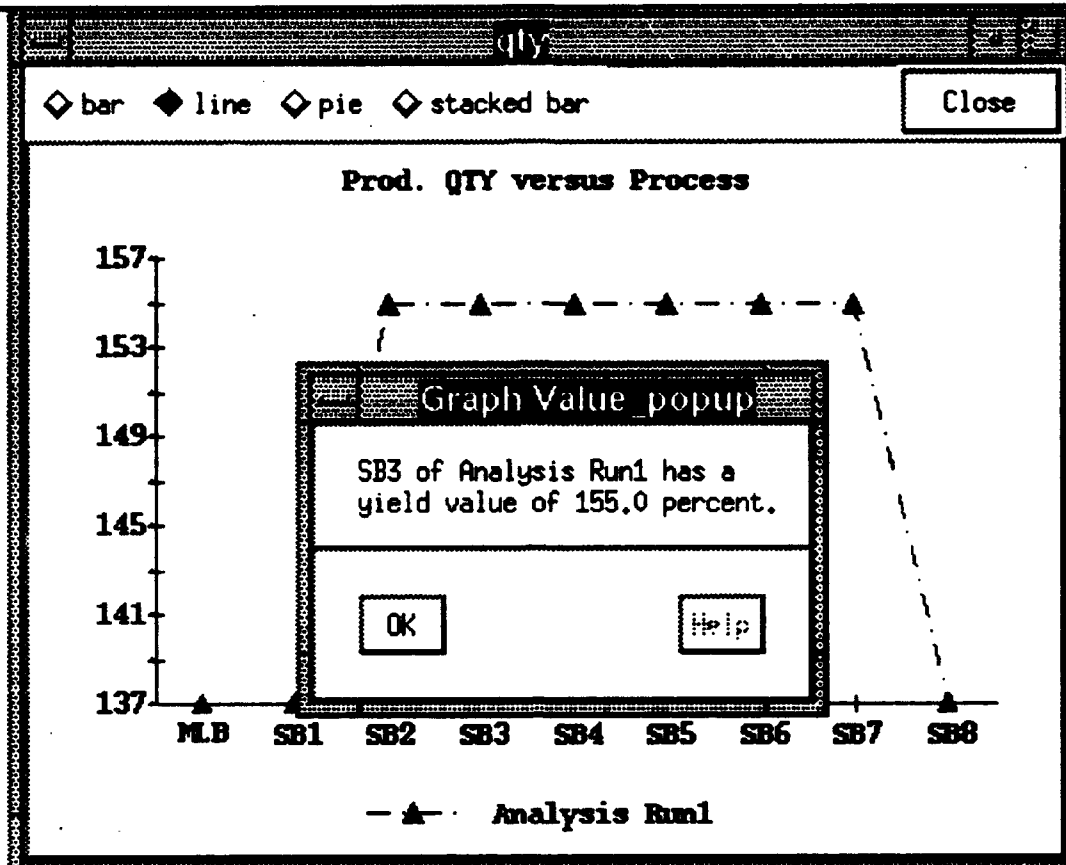


Figure III.3.3.3-3 Quantity Value for an Exact Point

III.3.4 Reports

The Reports button provides the means to generate reports for the results produced by each process participating in an analysis. This includes the ability to view process flows, yield & rework, and cost. A final summary report, identifying cost drivers, for each process contributing to a multi-process analysis for a given design database can also be generated. Figure III.3.4-1 is displayed when a user selects the advisor reports button. The user can then select the type of data that he/she wants in the output report.

☒ Process Flow
☐ Yield / Rework
☐ Costing
☐ Final Report

OK Cancel Help

Figure III.3.4-1 Analysis Reports Form

Provided below is a sample report generated from the Manufacturing Advisor based on the process flow and corresponding yield results for a PWB Fabrication process.

Fabrication Process Selection/Cost Estimation Report

MLB - layers 1, 14 OVERALL YIELD is 94 percent

<i>Opno</i>	<i>Description</i>	<i>Ideal(\$)</i>	<i>Actual(\$)</i>	<i>Rework(\$)</i>	<i>Yield</i>	<i>Rework</i>	<i># Units</i>
10	mark part no	0.123	0.12	0.00	100	0.000	137
30	oxide treat	1.111	1.11	0.00	100	0.000	137
40	bake panels	0.444	0.44	0.00	100	0.000	137
50	lay up	3.123	3.12	0.00	100	0.000	137
60	laminare	0.600	0.80	0.00	94	0.000	137
80	route excess	0.715	0.92	0.00	100	0.000	128
90	oxide strip	0.250	0.32	0.00	100	0.000	128
110	drill tooling	0.220	0.28	0.00	100	0.000	128
130	drill	12.123	15.12	1.23	92	0.005	128
160	electroless	0.661	0.66	0.00	100	0.000	117
170	copper panel	0.555	0.55	0.00	100	0.000	117
180	electrostrike	0.512	0.70	0.00	98	0.000	117

Fabrication Yield Analysis Report

MLB - layers 1, 14 OVERALL YIELD IS 94 percent

<i>Opno</i>	<i>Design Feature Description</i>	<i>Value</i>	<i>Scrap Per Feature</i>	<i>Opno Yield</i>
60	14 layers and 8 substrates	N/A	6.000	94
130	annular ring	8.00	8.000	92
180	aspect ratio	4.00	2.000	98

III.4 Modeler Window

The Process Modeler will provide manufacturing engineers with the ability to model the manufacturing processes of their products. The process model used in the MO system is designed as a hierarchical planning system. The hierarchical planning system is developed as a general purpose tree structure. The hierarchical tree consist of Manufacturing Activity nodes. Each Manufacturing Activity node consists of the following:

- Reasoning Logic - If these rules are satisfied, then the activity node is included in the total process analysis model.
- Manufacturing Data - There are three type of manufacturing data supported in the MO hierarchy model. The three data types are processes, operations, and steps. The data will be modeled by linking manufacturing processes to operations, and operations to steps. Each operation is annotated with its associated yield and rework rates.
- Resources - At each process, operation, or step node there is a list of resources attached. A resource is any facility, person, equipment, or consumable material used in the manufacturing process.
- Ordering - The children of a Manufacturing Activity node are defined with an imposed order of concurrent or sequential flow when building the model.

The Process Modeler provides functionality to create new manufacturing activity nodes and edit, copy, and delete existing nodes. Included in this functionality is a means to specify selection rules for the manufacturing activity nodes, define the manufacturing data (i.e. process, operation, or step) attached to the activity node, and identify the ordering for the children activities as either concurrent or sequential. Associated with each operation are scrap and rework rates. The Process Modeler window is shown in Figure III.4-1. A sample process model is displayed.

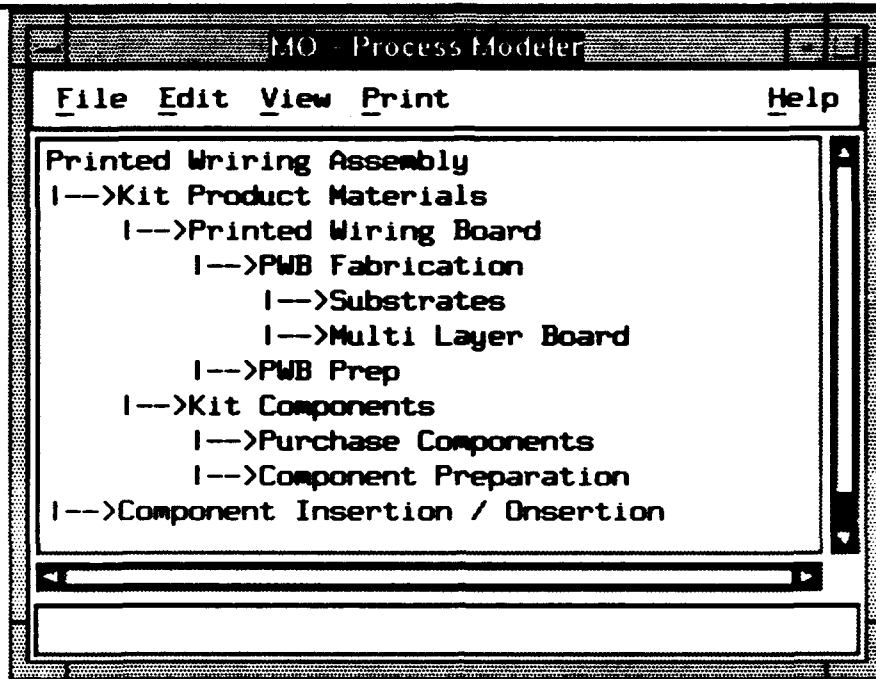


Figure III.4-1 Process Modeler Window

The user is provided the ability to create new process models and select, delete, and copy existing process models. These operations are done through the Process Model Selection window shown in Figure III.4-2 which is accessed by selecting the Models icon from the Process Model menu bar shown in Figure III.4-1.

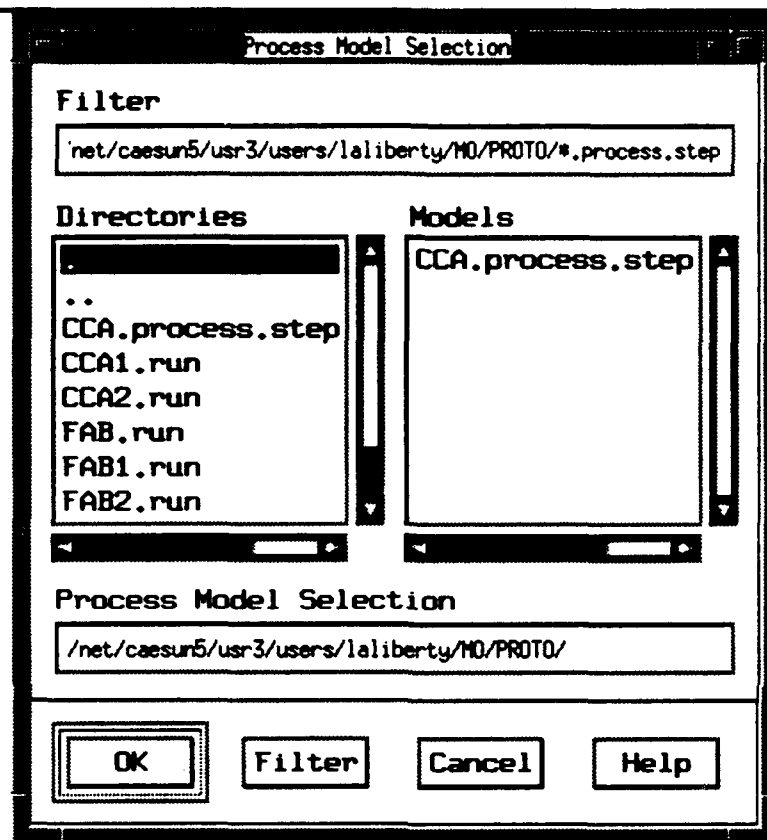
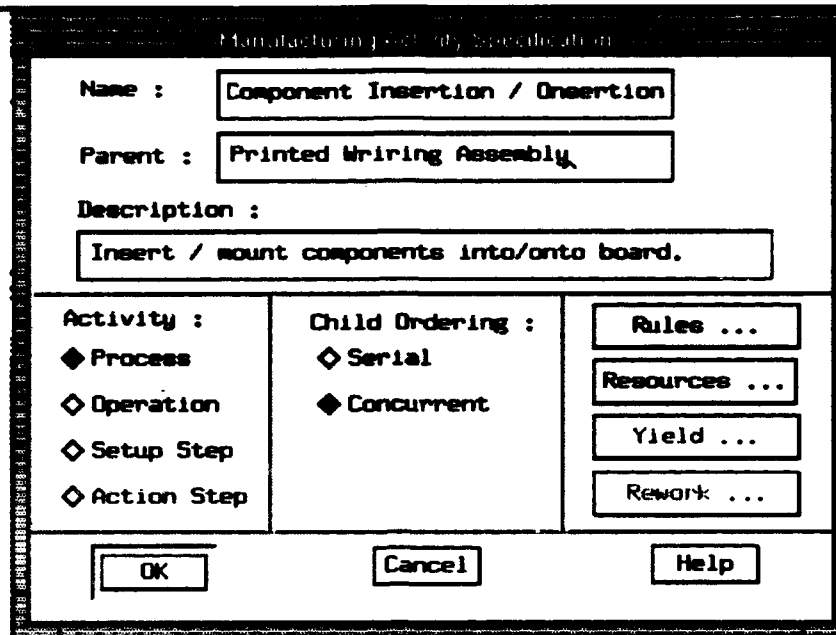


Figure III.4-2 Process Model Selection Window

III.4.1 Manufacturing Activity Node Definition

Defining a new process node will consist of selecting the Add icon from the Process Modeler Window menu bar and specifying the name of the node to the Manufacturing Activity Specification window shown in Figure III.4.1. The interface will then support activity data type selection (process, operation, or step), child ordering, selection rules, and resources.

Editing existing nodes will be accomplished by graphical selection of the desired node from the process modeler window (see Figure III.4-1) via the mouse. Once the activity has been selected, the Manufacturing Activity Specification window will be displayed with the data for the selected activity node loaded. Existing nodes are deleted by selecting the Delete icon and then the activity node to be deleted.



Manufacturing Activity Specification

Name :

Parent :

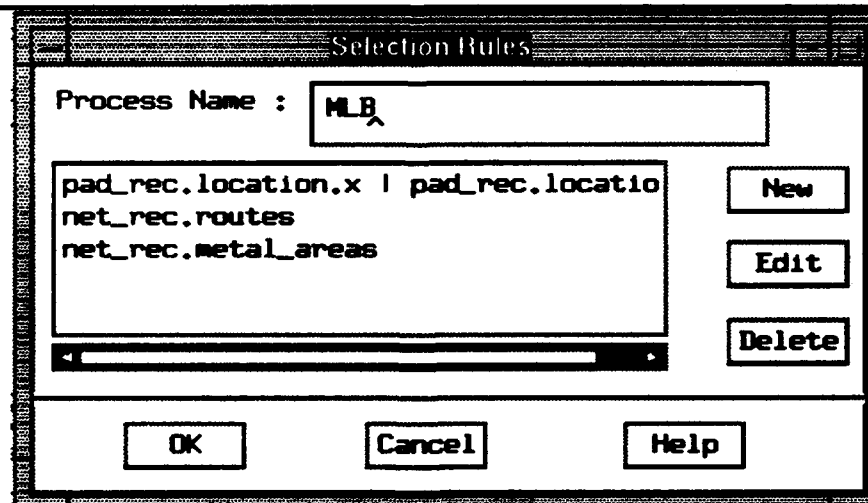
Description :

Activity : <input checked="" type="checkbox"/> Process <input type="checkbox"/> Operation <input type="checkbox"/> Setup Step <input type="checkbox"/> Action Step	Child Ordering : <input type="checkbox"/> Serial <input checked="" type="checkbox"/> Concurrent	<input type="button" value="Rules ..."/> <input type="button" value="Resources ..."/> <input type="button" value="Yield ..."/> <input type="button" value="Rework ..."/>
---	--	---

Figure III.4.1 Manufacturing Activity Specification Window

III.4.2 Selection Rules Definition

The window in Figure III.4.2-1 will support the creation, modification, and deletion of selection rules for an activity node. There will be an implicit OR between each of the rules in the list for an activity node, i.e., if one of the rules in the list is satisfied, then the node will be selected.



Selection Rules

Process Name :

Figure III.4.2-1 Selection Rules Window

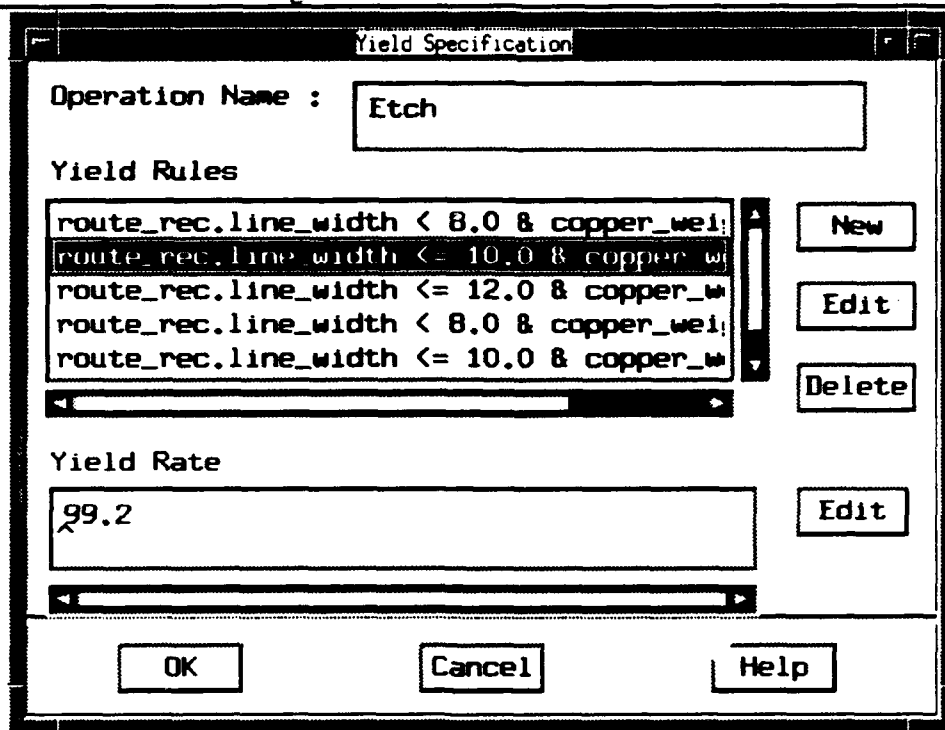
When either a new rule is to be defined or an existing one modified, the Rule Specification window shown in Figure III.4.2-2 will be presented. This window will also be utilized to specify scrap and rework rate rules and operation setup and operation run time rules.

Figure III.4.2-2 Rule Specification

III.4.2.1 Yield Rate Definition

Attached to every operation is a list of yield rates. A yield rate can be associated to a given entity attribute or a set of entity attributes. This is specified through an ordered list of rules and yield rates. The yield rate is established using the yield rate equation attached to the first yield rule in the ordered list that is satisfied. A user interface supporting this functionality is shown

in Figure III.4.2.1. The yield rules and the yield rates are specified through the Rule Specification Window shown in Figure III.4.2-2.



The image shows a software window titled "Yield Specification". It contains the following elements:

- Operation Name :** A text box containing the word "Etch".
- Yield Rules:** A list box containing five rules:
 - route_rec.line_width < 8.0 & copper_wel
 - route_rec.line_width <= 10.0 & copper w
 - route_rec.line_width <= 12.0 & copper_w
 - route_rec.line_width < 8.0 & copper_wel
 - route_rec.line_width <= 10.0 & copper_wVertical scroll bars are visible on the right of the list box.
- Yield Rate:** A text box containing the value "99.2".
- Buttons:** To the right of the rules list are three buttons: "New", "Edit", and "Delete". To the right of the yield rate text box is an "Edit" button. At the bottom of the window are three buttons: "OK", "Cancel", and "Help".

Figure III.4.2.1 Yield Specification Window

III.4.2.2 Rework Rate Definition

Attached to every operation is a list of rework rates. A rework rate can be associated for a given entity attribute or a set of entity attributes. This is specified through an ordered list of rules and rework rates. The rework rate is established using the rework rate equation attached to the first rework rule in the ordered list that is satisfied. A user interface supporting this functionality is shown in Figure III.4.2.2. The rework rules and rates is specified through the Rule Specification Window shown in Figure III.4.2-2. Also attached to each rework rule rate pair is a list of resources that is required to complete the rework activities. The resources used along with their associated setup and run time equations is specified using the Resource Utilization interface shown in Figure III.4.3-1.

Rework Specification

Operation Name : Etch

Rework Rules

- route_rec.line_width < 8.0 & copper_wel
- route_rec.line_width <= 10.0 & copper_w
- route_rec.line_width <= 12.0 & copper_w
- route_rec.line_width < 8.0 & copper_wel
- route_rec.line_width <= 10.0 & copper_w

New
Edit
Delete

Rework Rate

route_rec.line_width * 1.8976

Edit
Resources

OK Cancel Help

Figure III.4.2.2 Rework Specification Window

III.4.3 Resource Definition

For each process, operation, or step performed, a list of needed resources can be specified. When resources are utilized the amount of setup time and run time that is required for the resource must also be provided so that proper costing can be calculated. Figure III.4.3-1 shows the Resource Utilization interface that will allow the process modeler to construct the list of resources utilized by a process, operation, or step. The setup and run time equations are specified using the Rule Specification interface shown in Figure III.4.2-2.

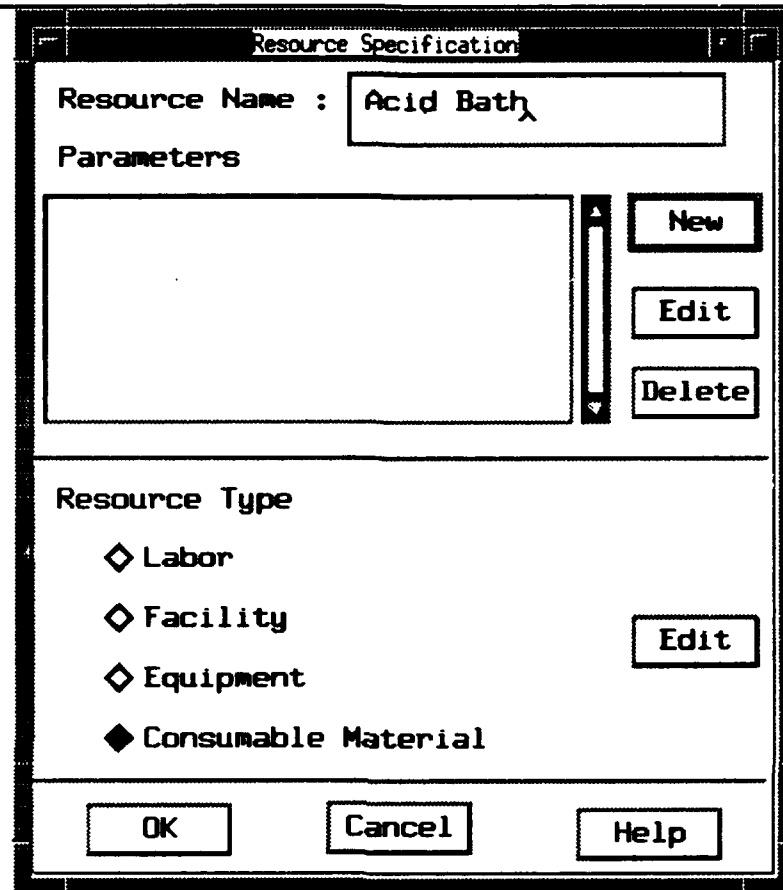
Figure III.4.3-1 Resource Utilization Window

Figure III.4.3-2 shows the Resources interface which lists all of the Resources that are currently stored in the process model. The interface supports creating new resources, and editing and deleting existing resources. To access the Resources interface the user would select the Resources icon in Figure III.4.1.

Figure III.4.3-2 Resource Window

The Resource Specification interface is shown in Figure III.4.3-3. This interface is used for specifying new resources and modifying existing ones. Attached to each resource is a list of

user definable parameters or attributes. Each resource falls into one of the following four categories: labor, facility, equipment, or consumable resource.

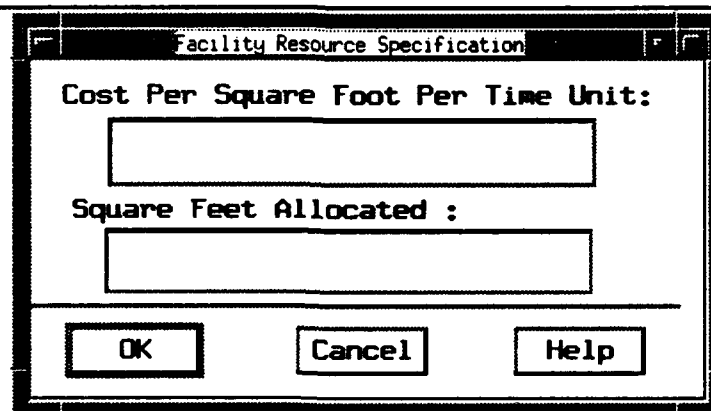


The 'Resource Specification' window contains the following elements:

- Title Bar:** Resource Specification
- Resource Name:** A text field containing 'Acid Bath'.
- Parameters:** A large empty text area for defining parameters.
- Buttons:** 'New', 'Edit', and 'Delete' buttons are positioned to the right of the Parameters text area.
- Resource Type:** A section with four radio button options:
 - ☐ Labor
 - ☐ Facility
 - ☐ Equipment
 - ☒ Consumable Material
- Edit Button:** An 'Edit' button is located to the right of the Resource Type options.
- Footer Buttons:** 'OK', 'Cancel', and 'Help' buttons are at the bottom of the window.

Figure III.4.3-3 Resource Specification Window

Figure III.4.3-4 shows the interface for specifying a facility resource.

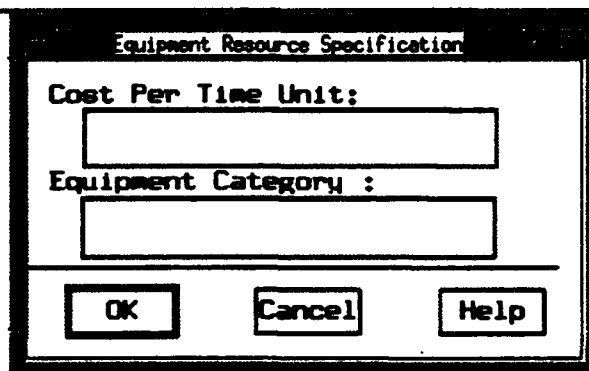


The 'Facility Resource Specification' window contains the following elements:

- Title Bar:** Facility Resource Specification
- Cost Per Square Foot Per Time Unit:** A text field for entering the cost rate.
- Square Feet Allocated:** A text field for entering the allocated area.
- Buttons:** 'OK', 'Cancel', and 'Help' buttons are at the bottom of the window.

Figure III.4.3-4 Facility Resource Specification Window

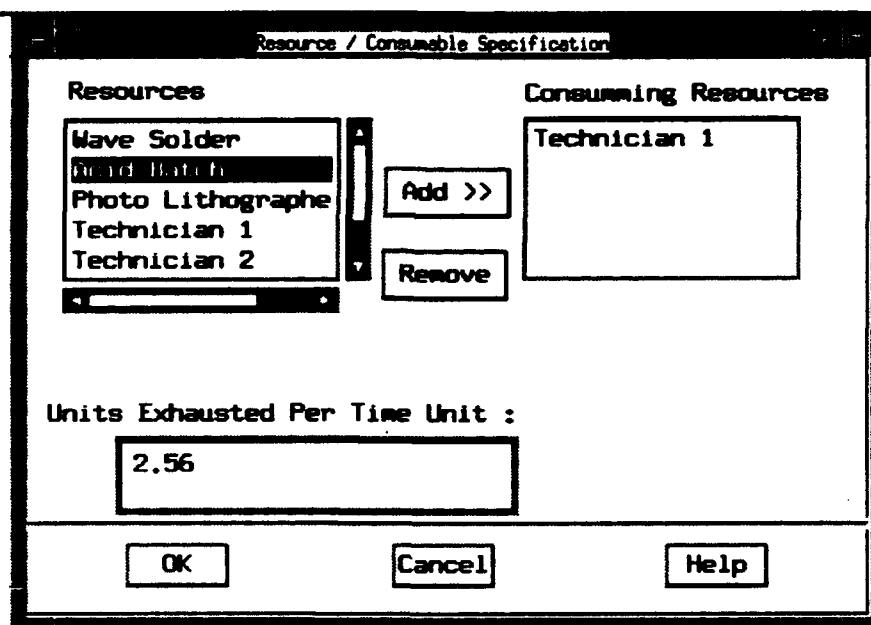
Figure III.4.3-5 shows the interface for specifying an equipment resource.



The dialog box is titled "Equipment Resource Specification". It contains two text input fields: "Cost Per Time Unit:" and "Equipment Category:". Below these fields are three buttons: "OK", "Cancel", and "Help".

Figure III.4.3-5 Equipment Resource Specification Window

An individual resource, consumable material pair is specified in the Resource/Consumable Specification Window shown in Figure III.4.3-6.



The dialog box is titled "Resource / Consumable Specification". It features two list boxes: "Resources" on the left and "Consuming Resources" on the right. The "Resources" list contains "Wave Solder", "Board Hatch", "Photo Lithographie", "Technician 1", and "Technician 2". The "Consuming Resources" list contains "Technician 1". Between the lists are "Add >>" and "Remove" buttons. Below the lists is a text input field labeled "Units Exhausted Per Time Unit :" with the value "2.56". At the bottom are "OK", "Cancel", and "Help" buttons.

Figure III.4.3-6 Resource/Consumable Specification Window

Figure III.4.3-7 shows the interface for specifying labor resources.

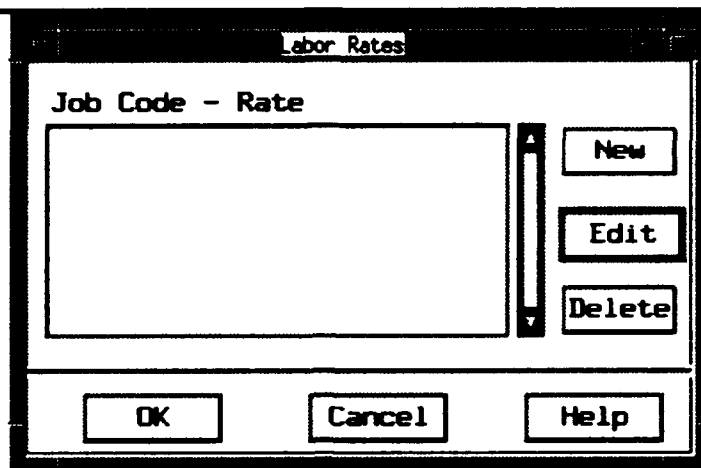


Figure III.4.3-7 Labor Resource Window

Figure III.4.3-8 shows the interface for specifying a labor rate resource.

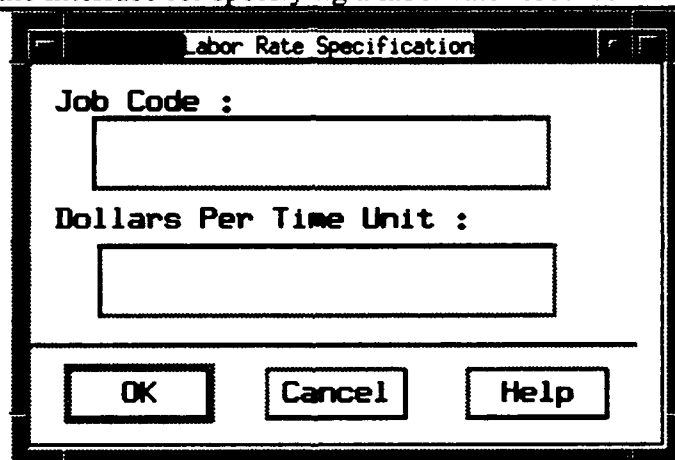


Figure III.4.3-8 Labor Rate Resource Specification Window

Distribution List

**DPRO-Raytheon
C/O Raytheon Company
Spencer Lab., Wayside Ave.
(one copy of each report)**

**Defense Advanced Research Projects Agency
ATTN: Defense Sciences Office; Dr. H. Lee Buchanan
Virginia Square Plaza
3701 N. Fairfax Drive
Arlington, VA. 22203-1714
(one copy of each report)**

**Defense Advanced Research Projects Agency
ATTN: Electronic Systems Technology Office; Capt. Nicholas J. Naclerio, USAF
Virginia Square Plaza
3701 N. Fairfax Drive
Arlington, VA. 22203-1714
(one copy of each report)**

**Defense Advanced Research Projects Agency
ATTN: Contracts Management Office; Mr. Donald C. Sharkus
Virginia Square Plaza
3701 N. Fairfax Drive
Arlington, VA. 22203-1714
(one copy of each report)**

**Defense Advanced Research Projects Agency
ATTN: OASB Library
Virginia Square Plaza
3701 N. Fairfax Drive
Arlington, VA. 22203-1714
(one copy of Final report only)**

**Defense Technical Information Center
Building 5, Cameron Station
ATTN: Selections
Alexandria, VA 22304
(two copies of each report)**